

MATLAB 仿真与应用系列丛书

详解 MATLAB 在统计与工程数据分析中的应用

张德丰 周 燕 主 编 雷小平 副主编

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书介绍 MATLAB 的基础知识、数据统计的基本原理、典型应用,以及用 MATLAB 进行工程数据处理与分析的基本方法。精选了科学和工程中常用的多个算法,采用 MATLAB 语言编程实现,并结合实例对算法程序进行验证和分析。具体内容包括 MATLAB 的基本知识、MATLAB 的程序设计及数值计算、MATLAB 的符号计算、数据分析与概率分布、统计分析图、方差分析、估计及假设检验、回归分析、数理统计的其他分析、工程数据分析中的应用等。

本书既可作为本科生和硕士研究生学习 MATLAB 的教材,也可作为科技人员使用 MATLAB 进行数据分析时的工具书或参考书,对从事程序开发人员也有一定的参考价值。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

详解 MATLAB 在统计与工程数据分析中的应用 / 张德丰, 周燕主编. —北京: 电子工业出版社, 2010.6
(MATLAB 仿真与应用系列丛书)

ISBN 978-7-121-10993-5

I. ①详… II. ①张… ②周… III. ①统计分析—计算机辅助计算—软件包, MATLAB IV. ①C812

中国版本图书馆 CIP 数据核字(2010)第 100237 号

责任编辑: 陈韦凯

特约编辑: 吴晓涛

印 刷:

装 订:

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×1092 1/16 印张: 23.25 字数: 595 千字

印 次: 2010 年 6 月第 1 次印刷

印 数: 4 000 册 定价: 45.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

前 言

MATLAB 和 Mathematica、Maple 并称为三大数学软件，它在数学类科技应用软件中的数值计算方面首屈一指。MATLAB 可以进行矩阵运算、绘制图形、实现算法、创建用户界面、连接其他编程语言的程序等，主要应用于工程计算、控制设计、信号处理与通信、图像处理、信号检测、金融建模设计与分析等领域。

本书以通俗易懂的形式，详细介绍了 MATLAB 的基础知识与各种运算，由浅入深系统地阐述了 MATLAB 语言的各种数据类型和基本编程方法，以简练和具有代表性的示例向读者演示了 MATLAB 的使用方法和操作技巧，为初识 MATLAB 的读者提供了有力的向导，使读者轻松跨入 MATLAB 的大门。

随着计算机的发展与普及，数理统计已成为处理信息、进行决策的重要理论和方法。在科学研究中，用数理统计方法从数据中获取信息和判别初步规律，往往成为重大科学发现的先导。数理统计是数学方法与实际相结合，应用最为广泛、最为重要的方式之一。因此，现代科研人员和工程技术人员都应该掌握数理统计的基础知识。MATLAB 是一套高性能的数值计算和可视化软件，它集矩阵运算、数值分析、信号处理和图形显示于一体，构成了一个界面友好、使用方便的用户环境，是实现数据分析与处理的有效工具。

本书介绍了 MATLAB、数据统计的基本原理、典型应用，以及用 MATLAB 进行实际工程数据处理与分析的基本方法。全书共分 10 章。第 1 章 MATLAB 的概述，包括 MATLAB 的简单介绍、MATLAB 操作界面、MATLAB 常用的函数等内容；第 2 章 MATLAB 的程序设计及数值计算，包括 MATLAB 程序结构、M 文件和 MATLAB 函数的调用与参数传递等内容；第 3 章 MATLAB 的符号计算，包括符号计算的基础、符号矩阵的生成、符号的基本运算等内容；第 4 章数据分析与概率分布，包括随机数的产生、随机数的使用等内容；第 5 章统计分析图，包括统计图、统计工序管理图等内容；第 6 章方差分析，包括单因素方差分析、双因素方差分析、多因素方差分析等内容；第 7 章估计及假设检验，包括参数估计、区间估计、假设检验等内容；第 8 章回归分析，包括一元线性回归分析、多元线性回归分析等内容；第 9 章数理统计的其他分析，包括聚类分析、判别分析、试验分析等内容；第 10 章工程数据分析中的应用，包括线性优化问题、非线性优化问题、二次规划问题等内容。

本书既可作为本科生和硕士研究生学习 MATLAB 语言的教材，也可作为科技人员使用 MATLAB 进行数据分析时的工具书或参考书，对程序开发人员也具有一定的参考价值。

为便于学习，本书免费提供部分程序的源代码，读者可登录 www.hxedu.com.cn（华信教育资源网）查找本书下载。

本书主要由张德丰、周燕和雷小平负责编写。参与图书编写及源程序校对、调试等工作的还有周灵、崔如春、李娅、栾颖、刘志为和周品等。

由于作者水平有限，本书虽然是在经多年使用和修改的讲稿基础上整理编写的，但书中一定还存在很多缺点和不足，恳请读者批评指正。

作 者
2010 年 3 月

目 录

第 1 章	MATLAB 的概述	(1)
1.1	MATLAB 的简单介绍	(1)
1.1.1	MATLAB 的发展史	(1)
1.1.2	MATLAB 的特点	(2)
1.1.3	MATLAB R2009a 新性能	(3)
1.2	MATLAB 操作界面	(4)
1.2.1	MATLAB 命令窗口	(4)
1.2.2	MATLAB 命令历史窗口	(5)
1.2.3	MATLAB 工作内存浏览器窗口	(5)
1.2.4	MATLAB 路径管理器窗口	(6)
1.2.5	MATLAB 工具栏	(6)
1.2.6	MATLAB 主菜单	(7)
1.3	MATLAB 常用的函数	(9)
1.3.1	环境命令	(9)
1.3.2	数组的函数	(9)
1.3.3	特殊变量和常数	(11)
1.4	一般矩阵表示法	(12)
1.4.1	数组与矩阵的概念	(12)
1.4.2	矩阵的建立	(13)
1.4.3	矩阵的拆分法	(14)
1.5	特殊矩阵表示法	(17)
1.6	矩阵的运算	(19)
1.6.1	矩阵的代数运算	(19)
1.6.2	矩阵关系运算	(25)
1.6.3	矩阵的逻辑运算	(27)
1.7	MATLAB 帮助系统	(28)
1.7.1	联机帮助系统	(28)
1.7.2	命令窗口查询帮助系统	(30)
第 2 章	MATLAB 的程序设计及数值计算	(33)
2.1	MATLAB 程序结构	(33)
2.1.1	顺序结构	(33)
2.1.2	分支结构	(34)
2.1.3	循环结构	(37)
2.2	M 文件	(40)
2.2.1	M 文件类型	(40)
2.2.2	M 文件的结构	(41)

2.2.3	M 文件的创建	(42)
2.3	MATLAB 函数的调用与参数传递	(42)
2.3.1	函数的调用	(42)
2.3.2	参数传递	(44)
2.4	MATLAB 的编程技巧	(47)
2.4.1	线性索引技巧	(47)
2.4.2	嵌套计算技巧	(47)
2.4.3	循环计算技巧	(49)
2.4.4	利用“:”和 end 技巧	(49)
2.4.5	使用全局变量技巧	(50)
2.4.6	使用例外处理机制技巧	(51)
2.4.7	倒序法技巧	(52)
2.4.8	向量法处理技巧	(52)
2.5	插值和拟合	(53)
2.5.1	一维插值	(53)
2.5.2	二维插值	(56)
2.5.3	高维插值	(58)
2.5.4	最小二乘拟合	(60)
2.5.5	多项式拟合	(64)
2.5.6	非线性拟合	(67)
第 3 章	MATLAB 的符号计算	(69)
3.1	符号计算的基础	(69)
3.1.1	符号计算的基本概念	(69)
3.1.2	符号表达式的创建	(71)
3.2	符号矩阵的生成	(72)
3.2.1	使用 sym 函数创建符号矩阵	(72)
3.2.2	将数值矩阵转化为符号矩阵	(72)
3.2.3	用创建子阵的方法创建符号矩阵	(72)
3.3	符号的基本运算	(73)
3.3.1	符号的代数运算	(73)
3.3.2	提取符号表达式中的分子与分母	(75)
3.4	矩阵的分解与化简	(76)
3.4.1	矩阵的特征值分解	(76)
3.4.2	矩阵的奇异值分解	(76)
3.4.3	矩阵的零列空间	(77)
3.4.4	因式分解	(78)
3.4.5	同类项合并	(78)
3.4.6	分式通分	(78)
3.5	符号微积分	(79)
3.5.1	符号极限	(79)

3.5.2	符号级数	(79)
3.5.3	符号微分	(80)
3.5.4	符号积分	(81)
3.5.5	符号积分变换	(82)
3.6	符号函数	(86)
3.6.1	复合函数的运算	(86)
3.6.2	反函数的运算	(87)
3.6.3	符号函数的可视化	(88)
3.7	符号方程的求解	(92)
3.7.1	代数方程的求解	(92)
3.7.2	微分方程的求解	(95)
第 4 章	数据分析与概率分布	(97)
4.1	随机数的产生	(97)
4.1.1	一般随机数生成	(97)
4.1.2	其他分布的随机函数	(98)
4.1.3	随机排序函数类型	(111)
4.1.4	概率密度函数	(112)
4.1.5	累积概率值	(122)
4.1.6	逆累积分布函数	(130)
4.2	随机数的使用	(137)
4.2.1	Galton 板实验	(137)
4.2.2	输赢问题	(139)
4.3	统计量的数字特征	(140)
4.3.1	数学期望与均值	(140)
4.3.2	数据比较	(142)
4.3.3	方差和标准差	(142)
4.3.4	累积和累和	(145)
4.3.5	协方差与相关系数	(146)
4.3.6	偏斜度和峰度	(147)
4.4	数据的属性与处理方法	(148)
4.4.1	评价指标矩阵与指标的无量纲化	(148)
4.4.2	客观性权向量建立的方法	(150)
4.4.3	综合评价的步骤	(151)
4.4.4	数据的属性与处理方法示例	(151)
第 5 章	统计分析图	(157)
5.1	统计图	(157)
5.1.1	样本图	(157)
5.1.2	误差图	(158)
5.1.3	交互图	(159)
5.1.4	概率图	(161)

5.1.5	其他统计图	(162)
5.2	统计工序管理图	(165)
5.2.1	工序图	(165)
5.2.2	密度图	(166)
5.2.3	密度、平均、均值图	(168)
5.3	频率分布表与频率直方图	(170)
5.4	非线性回归模型	(173)
5.4.1	非线性拟合	(173)
5.4.2	置信区间	(175)
5.5	主成分分析	(176)
5.5.1	巴特力特检验	(176)
5.5.2	PCA	(177)
5.6	实验设计	(180)
5.6.1	优化设计	(180)
5.6.2	因子设计	(182)
5.7	文件输入/输出	(183)
5.7.1	文件输入	(183)
5.7.2	文件输出	(184)
第 6 章	方差分析	(187)
6.1	单因素方差分析	(187)
6.1.1	单因素方差分析问题	(187)
6.1.2	单因素方差分析前提条件	(188)
6.1.3	单因素方差分析的步骤	(188)
6.1.4	单因素方差分析的 MATLAB 实现	(191)
6.2	双因素方差分析	(194)
6.2.1	双因素等重复试验的方差分析	(195)
6.2.2	双因素无重复试验的方差分析	(201)
6.3	多因素方差分析	(204)
6.4	多元方差分析	(207)
6.5	进一步讨论方差分析	(208)
6.5.1	把方差表输出到 Excel 中	(208)
6.5.2	方差表在图形窗口中的显示	(210)
第 7 章	估计及假设检验	(213)
7.1	参数的点估计	(213)
7.1.1	矩估计法	(213)
7.1.2	极大似然估计法	(214)
7.1.3	估计量的评选标准	(219)
7.2	区间估计	(221)
7.2.1	区间估计的基本概念	(221)
7.2.2	高斯—牛顿法的非线性最小二乘数据拟合	(223)



7.2.3	非线性模型的参数置信区间	(224)
7.2.4	非线性最小二乘预测置信区间	(226)
7.2.5	非线性拟合预测的交互图形	(226)
7.3	假设检验	(227)
7.3.1	假设检验的概念及步骤	(227)
7.3.2	总体参数的假设检验	(228)
7.4	单正态假设检验	(229)
7.4.1	单正态 U 检验	(229)
7.4.2	单正态 t 检验	(232)
7.5	双正态假设检验	(235)
7.6	正态性检验	(238)
7.7	总体参数检验	(243)
7.7.1	非正态总体样本的参数检验	(243)
7.7.2	总体分布的 χ^2 拟合检验	(244)
7.8	其他检验	(246)
7.8.1	秩和检验	(246)
7.8.2	中值检验	(247)
第 8 章	回归分析	(249)
8.1	概述	(249)
8.2	一元线性回归分析	(250)
8.2.1	一元线性回归分析数学模型	(250)
8.2.2	参数的最小二乘估计	(251)
8.2.3	回归显著性检验	(253)
8.2.4	回归方程的预测	(254)
8.2.5	一元线性回归函数介绍	(255)
8.2.6	一元线性回归分析的编程实现	(261)
8.3	多元线性回归分析	(266)
8.3.1	多元线性回归模型及矩阵表示	(267)
8.3.2	多元线性回归的显著性检验	(268)
8.3.3	β 的最小二乘估计	(270)
8.3.4	误差方差 σ^2 的估计	(270)
8.3.5	多元线性回归的预测	(271)
8.3.6	多元线性回归的实现	(272)
8.4	偏最小二乘回归分析	(277)
8.4.1	偏最小二乘回归分析	(277)
8.4.2	偏最小二乘回归方法的算法步骤	(278)
8.4.3	偏最小二乘回归方法分析	(280)
第 9 章	数理统计的其他分析	(285)
9.1	聚类分析	(285)
9.1.1	MATLAB 实现聚类分析	(285)



9.1.2	编程实现聚类分析	(291)
9.2	判别分析	(294)
9.2.1	MATLAB 实现判别分析	(294)
9.2.2	编程实现判别分析	(297)
9.3	试验分析	(300)
9.3.1	试验相关概述	(300)
9.3.2	试验分析的实现	(305)
9.4	正交实验设计	(308)
9.4.1	极差分析	(308)
9.4.2	方差分析	(311)
第 10 章	工程数据分析中的应用	(315)
10.1	工程优化问题的概述	(315)
10.2	线性优化问题	(316)
10.2.1	线性优化问题的基本知识	(316)
10.2.2	线性规划的 MATLAB 实现	(319)
10.3	非线性优化问题	(323)
10.3.1	有约束优化问题	(323)
10.3.2	无约束优化问题	(328)
10.4	二次规划问题	(333)
10.5	0-1 整数规划问题	(337)
10.5.1	0-1 整数规划概述	(337)
10.5.2	0-1 整数规划的实现	(338)
10.6	最大最小化问题	(339)
10.7	多元多目标函数优化	(341)
10.7.1	“半无限”多元函数优化	(341)
10.7.2	多目标函数优化	(343)
10.8	动态规划	(346)
10.8.1	动态规划的概念	(346)
10.8.2	逆序算法及 MATLAB 的实现	(348)
10.8.3	动态规划的应用	(352)
参考文献	(361)

第 1 章 MATLAB 的概述

1.1 MATLAB 的简单介绍

MATLAB 名字由 MATrix 和 LABoratory 两词的前三个字母组合而成。那是 20 世纪 70 年代后期的事：时任美国新墨西哥大学计算机科学系主任的 Cleve Moler 教授出于减轻学生编程负担的动机，为学生设计了一组调用 LINPACK 和 EISPACK 库程序的“通俗易懂”的接口，此即用 FORTRAN 编写的萌芽状态的 MATLAB。经几年的校际流传，在 Little 的推动下，由 Little、Moler、Steve Bangert 合作，于 1984 年成立了 MathWorks 公司，并把 MATLAB 正式推向市场。从这时起，MATLAB 的内核采用 C 语言编写，而且除原有的数值计算能力外，还新增了数据图视功能。

MATLAB 以商品形式出现后，仅短短几年，就以其良好的开放性和运行的可靠性，使原先控制领域里的封闭式软件包（如英国的 UMIST、瑞典的 LUND 和 SIMNON、德国的 KEDDC）纷纷淘汰，而改以 MATLAB 为平台加以重建。在时间进入 20 世纪 90 年代的时候，MATLAB 已经成为国际控制界公认的标准计算软件。到 90 年代初期，在国际上 30 几个数学类科技应用软件中，MATLAB 在数值计算方面独占鳌头，而 Mathematica 和 Maple 则分居符号计算软件的前两名。Mathcad 因其提供计算、图形、文字处理的统一环境而深受中学生欢迎。

MathWorks 公司于 1993 年推出 MATLAB 4.0 版本，从此告别 DOS 版。4.x 版在继承和发展其原有的数值计算和图形可视能力的同时，出现了以下几个重要变化：

（1）推出了 SIMULINK。这是一个交互式操作的动态系统建模、仿真、分析集成环境。它的出现使人们有可能考虑许多以前不得不做简化假设的非线性因素、随机因素，从而大大提高了人们对非线性、随机动态系统的认知能力。

（2）开发了与外部进行直接数据交换的组件，打通了 MATLAB 进行实时数据分析、处理和硬件开发的道路。

（3）推出了符号计算工具包。1993 年 MathWorks 公司从加拿大滑铁卢大学购得 Maple 的使用权，以 Maple 为“引擎”开发了 Symbolic Math Toolbox 1.0。MathWorks 公司此举加快结束了国际上数值计算、符号计算孰优孰劣的长期争论，促成了两种计算的互补发展新时代。

（4）构造了 Notebook。MathWorks 公司瞄准应用范围最广的 Word，运用 DDE 和 OLE，实现了 MATLAB 与 Word 的无缝连接，从而为专业科技工作者创造了融科学计算、图形可视、文字处理于一体的高水准环境。

1.1.1 MATLAB 的发展史

1997 年仲春，MATLAB 5.0 版问世，紧接着是 5.1、5.2，以及和 1999 年春的 5.3 版。与 4.x 相比，现今的 MATLAB 拥有更丰富的数据类型和结构、更友善的面向对象、更加快速精良的图形可视、更广博的数学和数据分析资源、更多的应用开发工具。诚然，到 1999 年底，Mathematica 也已经升到 4.0 版，它特别加强了以前欠缺的大规模数据处理能力。Mathcad 也赶在 2000 年到

来之前推出了 Mathcad 2000，它购买了 Maple 内核和库的部分使用权，打通了与 MATLAB 的接口，从而把其数学计算能力提高到专业层次。但是，就影响而言，至今仍然没有一个别的计算软件可与 MATLAB 匹敌。在欧美大学里，诸如应用代数、数理统计、自动控制、数字信号处理、模拟与数字通信、时间序列分析、动态系统仿真等课程的教科书都把 MATLAB 作为内容。这几乎成了 20 世纪 90 年代教科书与旧版书籍的区别性标志。在那里，MATLAB 是攻读学位的大学生、硕士生、博士生必须掌握的基本工具。在国际学术界，MATLAB 已经被确认为准确、可靠的科学计算标准软件。在许多国际一流学术刊物上（尤其是信息科学刊物），都可以看到 MATLAB 的应用。在设计研究单位和工业部门，MATLAB 被认作进行高效研究、开发的首选软件工具。如美国 National Instruments 公司信号测量、分析软件 LabVIEW，Cadence 公司信号和通信分析设计软件 SPW 等，或者直接建筑在 MATLAB 之上，或者以 MATLAB 为主要支撑。又如 HP 公司的 VXI 硬件，TM 公司的 DSP，Gage 公司的各种硬卡、仪器等都接受 MATLAB 的支持。之后陆续推出了几个改进和提高了的版本，2004 年 9 月正式推出 MATLAB Release14，即 MATLAB 7.0，其功能在原有的基础上又有了进一步的改进，2009 年 3 月推出 R2009a，它是目前 MATLAB 最新的版本。

1.1.2 MATLAB 的特点

MATLAB 的应用范围非常广，包括信号和图像处理、通信、控制系统设计、测试和测量、财务建模和分析以及计算生物学等众多应用领域。附加的工具箱（单独提供的专用 MATLAB 函数集）扩展了 MATLAB 环境，以解决这些应用领域内特定类型的问题。其功能特点主要有如下 6 点。

1. 大量引入图形用户界面

MATLAB 改变了过去单调依靠“在指令窗通过纯文本形指令进行各种操作”面貌，引入了许多让使用者一目了然的图形界面，如在线帮助的交互型界面 helpwin、管理工作内存的 workspace、交互式的路径管理界面 pathtool、指令窗显示风格设置界面等。它们的开启方式有工具条图标开启、选择菜单项开启、直接“文本式”指令开启。

2. 引入了全方位帮助系统

“临场”在线帮助，这些帮助内容，大多嵌附在 M 文件中，即时性强，反应速度快。它对求助内容的回答最及时准确。MATLAB 旧版就一直采用这种帮助系统，并深受用户欢迎。新版保留原功能的同时，还新增一个内容与之完全对应的图形界面 helpwin，加强了对用户的向导。综合型在线帮助文库 helpdesk：该文库以 HTML 超文本形式独立存在。整个文库按 MATLAB 的功能和核心内容编排，系统性强，且可以借助“超链接”方便地进行交叉查阅。但是，这部分内容偶而发生与真实 M 文件脱节的现象。完整易读的 PDF 文档：这部分内容与 HTML 帮助文库完全对应。PDF 文档不能直接从指令窗中开启，而必须借助 Adobe Acrobat Reader 软件阅读。这种文件的版面清楚、规范，适宜有选择地系统阅读，也适宜于制作硬拷贝。演示软件 demo：这是一个内容广泛的演示程序。MATLAB 一向重视演示软件的设计，因此无论 MATLAB 旧版还是新版，都随带各自的演示程序。只是，新版内容更丰富了。

3. M 文件编辑、调试的集成环境

新的编辑器有十分良好的文字编辑功能。它可采用色彩和制表位醒目地区分标识程序中不同功能的文字，如运算指令、控制流指令、注释等。通过编辑器的菜单选项可以对编辑器的文字、段落等风格进行类似 Word 那样的设置。从 5.2 版起，还新增了“变量现场显示”功能，只

要把鼠标放在变量名上 (Mouse over), 就能在现场显示该变量的内容。

在 5.x 以后版中, 调试器已经被图形化, 它与编辑器集成为一体。只需点动交互窗上的调试图标就可完成对程序的调试。

4. M 文件的性能剖析

调试器只负责 M 文件中语法错误和运行错误的定位, 而性能剖析指令 `profile` 将给出程序各环节的耗时分析报告。5.3 以后版都剖析指令的分析报告特别详细, 它将帮助用户寻找影响程序运行速度的“瓶颈”所在, 以便改进。

5. Notebook 新的安装方式

从 4.2c 版引入 Notebook 以来, 这种集文字、计算、图形于一体的“活”环境就深受用户赞赏。但直到 5.2 版, Notebook 的安装都是与 MATLAB 的安装同步进行的。这种安装方式的不便之处是: 一旦 Word 发生变动, 就必须把 MATLAB 全盘重装。5.3 以后的版都改变了这种局面, 它可以在 MATLAB 指令窗中“随时”进行安装 Notebook, 省时灵活。

6. MATLAB 环境可运行文件的多样化

旧版中, 用户可编制和运行的程序文件只有 M 脚本文件和 M 函数文件。5.x 以后版本都新增了产生伪代码 P 文件的 `pcode` 指令和产生二进制 MEX 文件的 `mex` 指令。较之 M 文件, 这两种文件的运行速度要快得多, 保密性也好。

1.1.3 MATLAB R2009a 新性能

Mathworks 公司于 2009 年 3 月发布了 MATLAB R2009a。相比以前版本而言, MATLAB R2009a 不仅包括 MATLAB 和 Simulink 的新特性, 同时还包含 81 个其他产品模块的升级和 bug 修正。

从 MATLAB R2009a 开始, MATLAB 和 Simulink 产品家族软件在安装后需要激活才能使用。MATLAB R2009a 将引入 License Center——在线 License 管理的工具。

MATLAB R2009a 新版本中, 产品模块进行了一些调整, MATLAB Builder for COM 的功能集成到 MATLAB Builder for .net 中去了, Financial Time Series Toolbox 的功能集成到 Financial Toolbox 中了。下面对 MATLAB 的新特点作简单介绍。

(1) MATLAB 产品家族新特性简要介绍如下:

① MATLAB 中采用先进的面向对象编程, 包括对类和对象、继承、方法、属性、事件和包的完全支持。

② Optimization Toolbox 中针对大量数据优化问题对内部点求解器和并行计算提供支持。

③ Financial Toolbox 均方差投资优化的线性互补程序。

④ Parallel Computing Toolbox 对 PBS Pro 和 TORQUE 规划的支持。

⑤ Statistics Toolbox 中交叉确认、特性选择、半随机数和并行最客服乘特性。

(2) Simulink 产品家族新特性简要介绍如下:

① Simulink 中重新设计的多平台库浏览器。

② Real-Time Workshop Embedded Coder 中生成对 AUTOSAR 兼容代码。

③ Embedded MATLAB 中 M-Lint 代码分析仪和 Simulink Design Verifier 对 Embedded MATLAB 语言子集函数生成代码进行检查。

④ Simulink Verification and Validation 提供对安全关键系统 IEC 61508 设计规则检查。

⑤ Simulink Fixed Point 提供对浮点模型的自动定点转换的指导意见。

- ⑥ Communication Blockset 针对调制、解调、编码和解码函数的定点支持。
- ⑦ Embedded IDE Link MU 作为新产品将 Simulink 模型生成代码并应用到 Green Hills MULTI 开发环境中。
- ⑧ 从 MATLAB R2008a 开始将不再支持 PowerPC 处理器上运行 Macintosh OS X 操作系统, 也不支持 Microsoft Windows 2000 操作系统。此外, 在 R2008a 中 15 个产品模块被重新命名。

1.2 MATLAB 操作界面

在 Windows 平台启动 MATLABR2009a 可以选择【开始】菜单下的【程序】子菜单下的【MATLAB R2009a】命令, 或双击在安装时自动在操作系统桌面创建的快捷方式, 或在 DOS 命令窗中直接输入“MATLAB”。

退出 MATLAB R2009a 时, 单击“关闭”按钮, 或在 MATLAB 桌面(非操作系统)选择“Exit MATLAB”(快捷方式 Ctrl+Q 或者 Alt+F4), 也可以在命令窗口(Command window)输入“quit”或“exit”函数, 即可退出 MATLAB R2009a。

MATLAB R2009a 启动后, 出现一个如图 1-1 所示的 MATLAB R2009a 桌面。桌面上包含一些 MATLAB 的工具。MATLAB 是一种命令式的语言, 用户可以通过界面、命令改变初始化的设置。

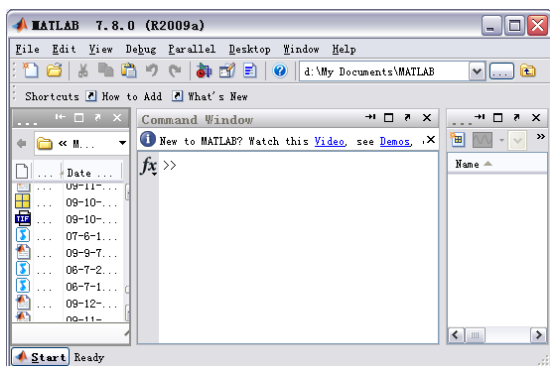


图 1-1 MATLAB 桌面

1.2.1 MATLAB 命令窗口

在默认设置下, “Command Window (命令窗口)” 自动显示于 MATLAB 界面右侧, 如果要单独显示命令窗口, 可单击【Desktop】菜单下【Desktop Layout】子菜单下的【Command Window Only】命令。

命令窗口是和 MATLAB 编辑器连接的主要窗口。`fx >>` 为运算提示符, 表示 MATLAB 处于准备状态。MATLAB 具有良好的交互性, 当在提示符后输入一段正确的运算式时, 只需按“Enter”键, 命令窗口就会直接显示运算结果。

【例 1-1】矩阵 $A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$ 的输入。

输入:

```
A=[1,4,7;2,5,8;3,6,9]
```

按回车键, 运行结果如图 1-2 所示。

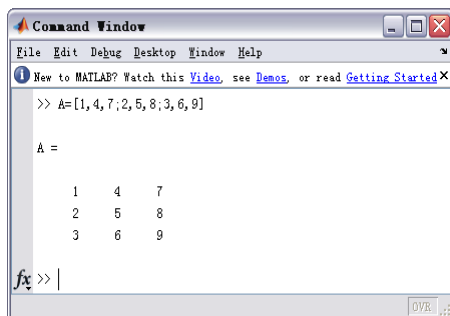


图 1-2 命令窗口显示结果

同时 MATLAB 的提示符 `>>` 不会消失, 这表明 MATLAB 继续处于准备状态。

1.2.2 MATLAB 命令历史窗口

在默认设置下, “Command History (命令历史)” 窗口自动显示于 MATLAB 界面左下侧, 用户可以通过单击【Desktop】菜单下的【Command History】命令, 调出或隐藏该窗口。

命令历史窗口显示用户在命令窗口中所输入的每条命令的历史记录, 并标明使用时间, 这样可以方便用户查询。如果用户要再次执行某条已经执行过的命令, 只需在命令历史窗口中双击该命令即可; 如果用户需要从命令历史窗口中删除一条或多条命令, 只需选中这些命令, 右击, 在弹出的快捷菜单中单击 “Delete Selection (删除选择)” 命令即可, 其窗口形式如图 1-3 所示。

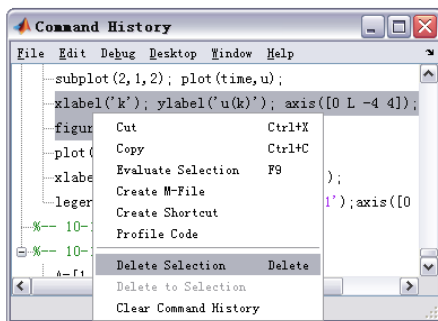


图 1-3 命令历史窗口

1.2.3 MATLAB 工作内存浏览器窗口

在默认设置下, “Workspace (工作内存浏览器)” 窗口自动显示于 MATLAB 界面左上侧, 用户也可以单击【Desktop】菜单下的【Workspace】命令, 调出或隐藏该窗口。

工作内存浏览器是 MATLAB 的重要组成部分, 例如, 表达式 `x=9` 产生了一个名为 `x` 的变量, 而且这个变量 `x` 被赋予 9 的值, 这个值就被存储在计算机的内存中。工作内存浏览器就是用来显示当前计算机内存中的 MATLAB 变量名称、数学结构、该变量的字节数及其类型, 如图 1-4 所示。

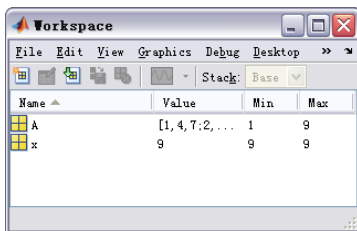


图 1-4 工作内存浏览器

1.2.4 MATLAB 路径管理器窗口

在默认设置下，“Current Directory（路径管理器）”窗口自动显示于 MATLAB 界面左上侧，用户也可以单击【Desktop】菜单下的【Current Directory】命令，调出或隐藏该窗口。

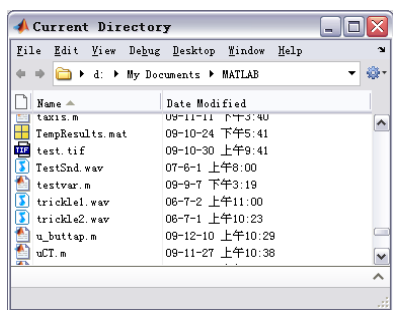


图 1-5 路径管理器窗口

“路径管理器”窗口显示当前用户工作所在的路径，窗口形式如图 1-5 所示。

当 Command Window 输入一条命令后，MATLAB 对该命令的基本搜索过程是，是否为内存变量→是否为内建函数→是否为当前目录上的 M 文件→是否为 MATLAB 路径上其他目录的 M 文件。如果在搜索路径上存在同名函数，则 MATLAB 仅发现搜索路径中的第一个函数，而其他同名函数不被执行。

使用以下命令可对当前的搜索路径进行操作：

- (1) Path：不加任何参数，显示当前搜索路径。
- (2) Path [路径名]：设置当前搜索路径，以前的搜索路径无效，如 Path D:\MATLAB R2009a\bin。
- (3) addpath D:\Mywork：向当前搜索路径中添加目录 D:\Mywork。
- (4) rmpath D:\Mywork：取消当前搜索路径中的目录 D:\Mywork。

此外，使用 Pathtool 命令或单击【File】菜单下的【Set Path】命令，可打开如图 1-6 所示的“Set Path”对话框，进行搜索路径的设置。

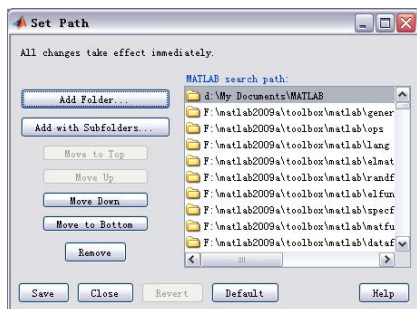


图 1-6 “Set Path”对话框

1.2.5 MATLAB 工具栏

在 MATLAB 桌面上，有许多操作选项和工具供用户使用，其中有些是 Windows 平台上常见的，有些是 MATLAB 所专有的。下面将简单介绍。

MATLAB 中的工具条如图 1-7 所示。

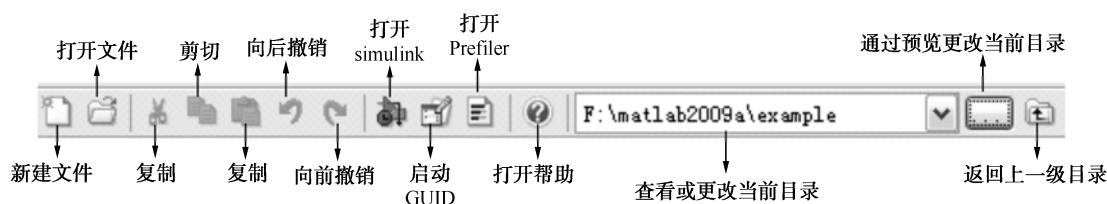


图 1-7 MATLAB 工具条

1.2.6 MATLAB 主菜单

MATLAB 桌面上的菜单使用方法和标准的 Windows 界面菜单一样，可以对 MATLAB 桌面上的内容进行操作。但 MATLAB 菜单会随着 MATLAB 桌面上分割窗体选择而发生变化，例如命令窗口（Command Window）处于活动状态（指 MATLAB 桌面当前操作的对象）和工作内存浏览器（Workspace）处于活动状态的菜单是不一样的。下面先针对命令窗口处于活动状态的 MATLAB 桌面菜单进行说明。其他情况读者可以参考相关资料。

1. File 菜单

New: 新建编辑、图形窗、MDL 文件、变量窗、GUI 等。

Open: 打开 MATLAB 所支持格式的文件。

Close Command Windows: 关闭命令窗口。

Import Data: 导入数据。

Save Workspace as: 将工作空间命令保存到文件中。

Page Setup: 打印设置位置。

Set Path: 调用路径浏览器。

PreferenceL: 调用 MATLAB 命令窗口环境设置界面。

Print: 打印。

Print Selection: 打印选定的内容。

Exit MATLAB: 退出 MATLAB。

2. Edit 菜单

Undo: 取消输入。

Redo: 重新输入。

Cut: 剪切。

Copy: 复制。

Paste: 粘贴。

Paste Special: 特殊粘贴（来自剪贴板的选择或文件）。

Select All: 全选。

Deletes: 删除。

Find: 寻找。

Find Files: 在指定的文件或路径中寻找。

Clear Command Window: 清除命令窗口中的显示。

Clear Command History: 清除命令历史窗中的显示。

Clear Workspace: 清除工作空间变量。

3. Debug 菜单

Step: 单步执行。

Step In: 进入执行。

Step Out: 退出执行。

Continue: 继续。

Clear Breakpoint in All Files: 清除所有断点。

Stop if Errors/Warning: 错误或警告停止。

Exit Debug Mode: 退出调试。

4. Parallel 菜单

Select Configuration: 选择配置。

Manage Configuration: 管理配置。

5. Desktop 菜单

Undock Command Window: 分离命令窗口（显示内容与当前活动窗有关）。

Desktop Layout: 桌面版面（标准、只有命令窗口、用户定义等）。

Save Layout: 保存当前面板。

Organize Layout: 组织面板。

Command Window: 显示或隐藏命令窗口。

Command History: 显示或隐藏命令历史窗口。

Command Directory: 显示或隐藏工作空间浏览器。

Help: 显示或隐藏帮助。

Profiler: 显示或隐藏性能分析器。

Editor: 显示或隐藏编辑器。

Figure: 显示或隐藏图形显示窗口。

Web Browser: 打开网络浏览器。

Array Editor: 打开数组编辑器。

File and Directory Comparisons: 文件和目录比较。

Filter Visualization Tool 过滤器可：视化工具。

Toolbar: 显示或隐藏工具栏。

Titles: 显示或隐藏窗体标题。

6. Window 菜单

Close All Documents: 关闭所有的文档（MATLAB 支持的文件）。

Next Tool: 显示下一个工具。

Previous Tool: 显示前一个工具。

Next tab: 显示下一个标签。

Previous tab: 显示前一个标签。

0 Command Window: 默认命令窗口。

1 Command History: 与当前打开的工具有关，这是默认桌面选择。

2 Current History: 有效的工具（即让其处于活动状态）。

3 Workspace: 工作空间浏览器。

7. Help 菜单

Product Help: 产品帮助。

Function Brower: 函数浏览器。

Using The Desktop: 打开桌面帮助。

Using The Command Window: 打开命令窗口帮助。

Web Recources: 网上资源。

Check for Updates: 检测更新。

Licensing: 协议许可。

Terms of use: 使用条款。

Patents: 使用专利。

Demos: 打开演示窗口。

About MATLAB: 显示 MATLAB 的版本及用户登记信息。

1.3 MATLAB 常用的函数

1.3.1 环境命令

1. clear 函数

clear 函数用来清除变量或函数。其主要调用格式如下:

clear: 清除工作空间中的所有变量。

clear variables: 清除工作空间中的所有变量。

clear global: 清除所有的全局变量。

clear functions: 清除所有已编辑过的 m 函数和 mex 函数。

clear all: 清除所有的变量、全局变量、函数和 mex 连接。在命令提示符下, 还可以清除 Java 包的输入列表。

clear import: 在命令提示符下, 清除 Java 包的输入列表; 这时不能清除函数。

clear classes: 和 clear all 类似, 但该命令把类的定义也清除了。

clear var1 var2 ...: 清除指定的变量, 可以用通配符 “*” 来清除所有含有相同字母的变量。

例如 clear x *清除工作空间中所有以 “x” 开头的变量; 如果是全局变量, 只从工作空间中清除, 在声明它的函数中仍然保留。

clear global x: 完全清除全局变量 x。

clear fun: 清除指定的函数 fun。

2. clc 函数

clc 函数用来清除命令窗口, 并把光标置于命令窗口的左上角。

用 clc 函数, 只清除命令窗口中所有显示的内容, 并不清除工作空间窗口中变量。

1.3.2 数组的函数

1. size 函数

size 函数可以获得数组的大小。其主要调用格式如下:

d = size(X): 若 x 是 m×n 矩阵, 返回两个元素的行向量 d=[m, n], 分别代表矩阵 x 的行数

m 和列数 n; 若 x 是 k 维数组, 则返回一个 k 维行向量, 它的每一个分量是每一维的维数的长度。

[m,n] = size(X): 返回矩阵 x 的行数 m 和列数 n。这是这一函数最常用的一种调用格式。

m = size(X,dim): 返回数组 x 的指定维 dim 的长度。

[d1,d2,d3,...,dn] = size(X): 返回数组 x 的第一维长度 m1; 第二维长度 m2; ...; 第 k 维长度 mk。如果 k 不等于数组的维数 n, 当 k>n 时, 从第 n+1 维开始到第 k 维返回值全都赋值为 1; 当 k<n 时, 第 k 维返回值是数组从第 k 维到第 n 维的各维数长度的乘积。

【例 1-2】size 函数示例。

```
>> m = size(rand(2,3,4),2)
m =
     3
>> d = size(rand(2,3,4))
d =
     2     3     4
>> [m,n,p] = size(rand(2,3,4))
m =
     2
n =
     3
p =
     4
```

2. length 函数

length 函数为获得数组的长度。其调用格式如下:

n=length(x): 返回向量 x 的长度; 对非空数组, 它与 max(size(x))等同, 返回的是各维长度中的最大值; 如果是空数组则返回 0。

【例 1-3】length 函数示例。

```
>> x = ones(1,8);
n = length(x)
n =
     8
>> x = rand(2,10,3);
n = length(x)
n =
    10
```

3. disp 函数

disp 函数为显示命令窗口中的内容。其调用格式如下:

disp(x): 在命令窗口中显示变量 x 的内容, 但不显示变量名。

【例 1-4】disp 函数示例。

```
>> disp('      Corn      Oats      Hay')
disp(rand(5,3))
```

运行程序, 显示如下:

Corn	Oats	Hay
0.1818	0.5797	0.3510
0.2638	0.5499	0.5132
0.1455	0.1450	0.4018

0.1361	0.8530	0.0760
0.8693	0.6221	0.2399

4. format 函数

format 函数用来控制数据显示的格式。在 MATLAB 中，存储和计算都是以双精度数组进行的，可以用 format 命令来控制数据显示的格式，其主要调用格式如下：

format：默认设置。

format short：5 位定点表示，是默认设置。

format long：15 位定点表示。

format short e：5 位浮点表示。

format long e：15 位浮点表示。

format short g：自动选择 5 位定点与 5 位浮点中最佳的表示格式。

format long g：自动选择 15 位定点与 15 位浮点中最佳的表示格式。

format hex：十六进制格式表示。

format +：用符号“+”、“-”和空格来表示矩阵中的各元素实部，各元素的虚部忽略。

format rat：用分数近似表示小数。

format compact：压缩表示。显示时，变量与数据之间不留空行。

format loose：松散表示。显示时，变量与数据之间留有空行。这里是默认设置。

format 函数只是控制数组在命令窗口中的显示形式，并不改变数组的存储方式，也就是说，不论在命令窗口中显示的是 5 位，还是 15 位，其数据都是以双精度数组形式存储的。

【例 1-5】format 函数示例。

```
>> format long
>> pi
ans =
    3.141592653589793
>> get(0,'format')
ans =
long
>> format('short','e')
>> pi
ans =
    3.1416e+000
```

1.3.3 特殊变量和常数

在 MATLAB 内有许多特殊的变量和常量，简单介绍如下：

(1) ans：默认变量，用来自动存储未定义的变量输出。例如：

```
>> a=[1 2 3;4 5 6];
>> b=[3 6 9;2 7 8];
>> a+b
ans =
     4     8    12
     6    12    14
```

(2) eps：计算误差。

d = eps(X)：计算 X 变量的误差。

eps('double'): 计算 X 变量的误差为 double 型。

eps('single'): 计算 X 变量的误差为 single 型。

【例 1-6】eps 函数示例。

```
>> eps(1/2)
ans =
    1.1102e-016
>> eps(1)
ans =
    2.2204e-016
>> eps(2)
ans =
    4.4409e-016
>> eps(realmax)
ans =
    1.9958e+292
>> eps(0)
ans =
    4.9407e-324
```

(3) realmax: 最大的浮点数, 超过它就产生浮点溢出。

(4) realmin: 最小的浮点数, 小于它就表示成 0。

(5) inf: 无穷大。

(6) NaN 或 nan: 不是一个数值、无效数值, 如 “0/0”、“inf-inf” 等。

(7) pi: 圆周率 $\pi=3.14159$ 。

(8) i,j: 虚数单位。例如, $4+5i$ 表示复数 “ $4+5i$ ”, $2-7j$ 表示复数 “ $2-7j$ ”。

1.4 一般矩阵表示法

1.4.1 数组与矩阵的概念

MATLAB 是 “Matrix Laboratory” 之意, 即是矩阵实验室。MATLAB 最初是为解算线性代数的问题而开发的, MATLAB 以矩阵作为基本的运算单元。矩阵是在线性代数中定义的。

线性代数中矩阵是这样定义的: 有 $m \times n$ 个数 $a_{ij}(i=1,2,\cdots,m;j=1,2,\cdots,n)$ 的数组将其排成如下格式 (用方括号括起来):

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

此表作为整体, 将它当做一个抽象的量称为矩阵, 且是 m 行 n 列的矩阵。横向每一行所有元素依次序排列则为行向量; 纵向每一列所有元素依次序排列则为列向量。请特别注意, 数组用方括号括起来后已作为一个抽象的特殊量——矩阵。在线性代数中, 矩阵有特定的数学函义, 并且有其自身严格的运算规则。矩阵概念是线性代数范畴内特有的。

在 MATLAB 中，定义了矩阵运算规则及其运算符。MATLAB 中的矩阵运算规则与线性代数中的矩阵运算规则相同。

数组（Array）是由一组复数排列的长方形阵列（而实数可视为复数的虚部为零的特例）。对于发展了的 MATLAB，在线性代数范畴之外，数组也是进行数值计算的基本处理单元。一行多列的数组是行向量；一行多列的数组就是列向量；数组可以是二维的“矩形”，也可以是三维的，甚至还可以是多维的。多行多列的“矩形”数组与数学中的矩阵从外观形式与数据结构上看，没有什么区别。

在 MATLAB 中，也定义了一套数组运算规则及其运算符，但数组运算是 MATLAB 软件所定义的规则，规则是为了管理数据方便、操作简单、指令形式自然、程序简单易读与运算高效。在 MATLAB 中的大量数值计算是以数组形式进行的。而在 MATLAB 中凡是涉及线性代数范畴的问题，其运算则是以矩阵作为基本的运算单元。

MATLAB 既支持数组的运算，也支持矩阵的运算。但在 MATLAB 中，数组与矩阵的运算却有很大的差别。在 MATLAB 中，数组的所有运算都是对运算数组中的每个元素平等地执行同样的操作。矩阵运算是从把矩阵整体当作一个特殊的量这个基点出发，依照线性代数的规则来描述的运算。

1.4.2 矩阵的建立

1. 直接输入法

最简单的建立矩阵的方法是从键盘直接输入矩阵的元素。具体方法如下：将矩阵的元素用方括号括起来，按矩阵的行的顺序输入各元素，同一行的各元素之间用空格或逗号分隔，不同行的元素之间用分号分隔。例如，输入命令：

```
>> X=[1 2 3;4 5 6;7 8 9]
X =
     1     2     3
     4     5     6
     7     8     9
```

这样，在 MATLAB 的工作空间中就建立了一个矩阵 X，以后就可以使用矩阵 X。

2. 利用 M 文件建立矩阵

对于比较大且比较复杂的矩阵，可以为它专门建立一个 M 文件。下面通过一个简单例子来说明如何利用 M 文件创建矩阵。

【例 1-7】利用 M 文件建立 mydatat 矩阵。

(1) 启动有关编辑程序或 MATLAB 文本编辑器，并输入待建矩阵：

```
mydatat=[11 12 13 14 15 16 17 18 19;
          21 22 23 24 25 26 27 28 29;
          31 32 33 34 35 36 37 38 39];
```

(2) 把输入的内容存盘（设文件名为 matrix.m）。

(3) 在 MATLAB 命令窗口中输入 matrix，即运行该 M 文件，就会自动建立一个名为 mydatat 的矩阵，可供以后使用。

3. 建立大矩阵

大矩阵可由方括号中的小矩阵建立起来。例如：

```
>> a=[1 2 3;4 5 6;7 8 9];
```

```
b=[a,eye(size(a));ones(3),a]
```

```
b =
```

```

1     2     3     1     0     0
4     5     6     0     1     0
7     8     9     0     0     1
1     1     1     1     2     3
1     1     1     4     5     6
1     1     1     7     8     9

```

其中：eye(3)返回 3×3 单位矩阵；ones(3)返回 3×3 全 1 矩阵。

4. 用冒号生成法

用直接输入的方式生成向量，既费力，又容易出错。在 MATLAB 中，可以用冒号“:”来生成等差形式的向量。其生成规则如下：

```
a=x1:step:x2
```

其中：x1 是初值；x2 是终值；step 是步长增量（即公差）。

【例 1-8】用冒号生成法示例。

```
>> a=1:0.2:2
```

运行程序，输出如下：

```
a =
```

```
1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
```

如果步长为 1，可以省略。

【例 1-9】步长为 1 的示例。

```
>> a=2:8
```

运行程序，输出如下：

```
a =
```

```
2     3     4     5     6     7     8
```

如果终值与初值的差不是步长的整数倍，得到的向量最后一个数值会小于终值，而不是等于终值。

1.4.3 矩阵的拆分法

1. 矩阵元素

MATLAB 允许用户对一个矩阵的单个元素进行赋值和操作。例如，如果想将矩阵 A 的第 3 行第 2 列的元素赋为 200，则可以通过下面语句来完成：

```
A(3,2)=200
```

这时将只改变元素的值，而不影响其他元素的值。如果给出的行下标或列下标大于原来矩阵的行数或列数，则 MATLAB 将自动扩展原来的矩阵，并将扩展后未赋值的矩阵元素置为 0。例如：

```
>> A=[2 5 9;1 4 7];
```

```
A(4,5)=9
```

```
A =
```

```

2     5     9     0     0
1     4     7     0     0
0     0     0     0     0
0     0     0     0     9

```

在 MATLAB 中，也可以采用矩阵元素的序号来引用矩阵元素。矩阵元素的序号就是相应

元素在内存中的排列顺序。矩阵元素按列编号，先第一列，再第二列，依次类推。例如：

```
>> A=[2 5 8;1 3 9];
A(3)
ans =
    5
```

显然，序号（Index）与下标（Subscript）是一一对应的，以 $m \times n$ 矩阵 A 为例，矩阵元素 $A(i,j)$ 的序号为 $(j-1)*m+i$ 。其相互转换关系也可利用 `sub2ind` 和 `ind2sub` 函数求得。例如：

```
>> sub2ind(size(A),1,2)
ans =
    3
>> [i,j]=ind2sub(size(A),3)
i =
    1
j =
    2
```

其中：`size(A)`函数返回包含两个元素的向量，分别是矩阵 A 的行数和列数。相关的函数有：`length(A)`给出行数和列数中的较大者，即 `length(A)=max(size(A))`；`ndims(A)`给出 A 的维数。

`reshape(A, m, n)`函数在矩阵总元素保持不变的前提下，将矩阵 A 重新排列成 $m \times n$ 的二维矩阵。例如：

```
>> x=[11 22 33 44 55 66 77 88 99 100 110 111]; %产生有 12 个元素的行向量 x
y=reshape(x,3,4) %利用向量 x 建立 3×4 矩阵 y
y =
    11    44    77   100
    22    55    88   110
    33    66    99   111
```

注意，在 MATLAB 中，矩阵元素按列存储，即首先存储矩阵的第一列元素，然后存储第二列元素……一直到矩阵的最后一列元素。`reshape` 函数只是改变原矩阵的行数和列数，即改变其逻辑结构，但并不改变原矩阵元素个数及它的存储结构。例如，再针对上面建立的矩阵 y ，执行命令：

```
>> newy=reshape(y,2,6)
newy =
    11    33    55    77    99   110
    22    44    66    88   100   111
```

2. 矩阵拆分

1) 利用冒号表达式获得子矩阵

(1) $A(:,j)$ 表示取 A 矩阵的第 j 列全部元素； $A(i,:)$ 表示 A 矩阵第 i 行的全部元素； $A(i,j)$ 表示取 A 矩阵第 i 行、第 j 列的元素。

(2) $A(i:i+m,:)$ 表示取 A 矩阵第 $i \sim i+m$ 行的全部元素； $A(:,k:k+m)$ 表示取 A 矩阵第 $k \sim k+m$ 列的全部元素； $A(i:i+m,k:k+m)$ 表示取 A 矩阵第 $i \sim i+m$ 行内，并在第 $k \sim k+m$ 列中的所有元素。例如：

```
>> A=[1 2 3 4 5;6 7 8 9 10;11 12 13 14 15;16 17 18 19 20]
A =
     1     2     3     4     5
     6     7     8     9    10
```



```

11    12    13    14    15
16    17    18    19    20
>> A(2:3,4:5)
ans =
     9     10
    14     15

```

又如:

```

>> A(2:3,1:2:5)
ans =
     6     8    10
    11    13    15

```

(3) $A(:)$ 将矩阵 A 每一列元素堆叠起来, 成为一个列向量, 而这也是 MATLAB 变量的内部储存方式。例如:

```

>> A=[22 33 44;55 66 77]
A =
    22    33    44
    55    66    77
>> B=A(:)
B =
    22
    55
    33
    66
    44
    77

```

在这里, $A(:)$ 产生一个 6×1 的矩阵, 等价于 $\text{reshape}(A, 6, 1)$ 。

此外, 还可利用一般向量和 end 运算符等来表示矩阵下标, 从而获得子矩阵。 end 表示某一维的末尾元素下标。例如:

```

>> A=[1 2 3 4 5;6 7 8 9 10;11 12 13 14 15;16 17 18 19 20];
A(end,:);           %取 A 最后一行元素
A([1,4],3:end)      %取 A 第 1,4 两行中第 3 列到最后一列的元素
ans =
     3     4     5
    18    19    20

```

2) 利用空矩阵删除矩阵的元素

在 MATLAB 中, 定义 $[]$ 为空矩阵。给变量 X 赋空矩阵的语句为 $X=[]$ 。注意, $X=[]$ 与 $\text{clear } X$ 不同, clear 是将 X 从工作空间中删除, 而空矩阵则存在于工作空间, 只是维数为 0。

将某些元素从矩阵中删除, 采用将其置为空矩阵的方法就是一种有效的方法。例如:

```

>> A=[1 2 3 4 5 6;7 8 9 10 11 12;13 14 15 16 17 18];
A(:,[2 4])=[]

```

其中第二条命令将删除 A 的第二列和第四列元素。输出如下:

```

A =
     1     3     5     6
     7     9    11    12
    13    15    17    18

```

1.5 特殊矩阵表示法

1. eye: 产生单位矩阵

其调用格式如下:

A=eye(n): 返回一个 $n \times n$ 阶单位矩阵。

A=eye(m, n): 返回一个 $m \times n$ 阶单位矩阵, 或用 **A=eye([m n])**。

A=eye(size(B)): 返回一个大小与矩阵 **B** 一样的单位矩阵。

说明: 其主对角线元素为 1, 其他元素均为 0。

【例 1-10】eye 函数用法示例。

```
>> A=eye(3,4)
```

```
A =
```

```
    1    0    0    0
    0    1    0    0
    0    0    1    0
```

2. zeros: 产生零矩阵或数组

其调用格式如下:

A=zeros(n): 返回一个 $n \times n$ 阶零矩阵。

A=zeros(m, n): 返回一个 $m \times n$ 阶零矩阵, 或用 **A=zeros([m n])**。

A=zeros(d1, d2, d3, ...): 返回一个维数为 $d1 \times d2 \times d3 \times \dots$ 的所有元素为 0 的数组。

A=zeros(size(B)): 返回一个大小与 **B** 一样的零矩阵。

说明: 矩阵或数组所有元素为 0。

【例 1-11】zeros 函数用法示例。

```
>> Z=zeros(3)
```

```
Z =
```

```
    0    0    0
    0    0    0
    0    0    0
```

```
>> A=zeros(3,4,5)
```

```
A(:,:,1) =
```

```
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

```
A(:,:,2) =
```

```
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

```
A(:,:,3) =
```

```
    0    0    0    0
    0    0    0    0
    0    0    0    0
```

```
A(:,:,4) =
```

```
    0    0    0    0
    0    0    0    0
```

```

0      0      0      0
A(:,5) =
0      0      0      0
0      0      0      0
0      0      0      0

```

3. rand: 可以生成 (0, 1) 区间上均匀分布的随机阵

其调用格式如下:

r = rand(n): 生成 $n \times n$ 的随机阵。

r = rand(m,n): 生成 $m \times n$ 的随机阵。如果 $m=1$, 则生成行向量; 若 $n=1$, 则生成列向量。此种调用的另一种方式是: **r=randn([m, n])**, 其效果和 **r=randn(m, n)** 一样。

r=rand(m,n,p,...): 生成 $m \times n \times p \times \dots$ 的随机多维数组。

r = rand(size(A)): 生成与矩阵 A 行列数相同的随机阵。

r=rand: 生成一个随机数。

【例 1-12】rand 函数用法示例。

```

>> r=rand(4,5)

r =

    0.8147    0.6324    0.9575    0.9572    0.4218
    0.9058    0.0975    0.9649    0.4854    0.9157
    0.1270    0.2785    0.1576    0.8003    0.7922
    0.9134    0.5469    0.9706    0.1419    0.9595

>> r=rand(4,4)

r =

    0.6557    0.6787    0.6555    0.2769
    0.0357    0.7577    0.1712    0.0462
    0.8491    0.7431    0.7060    0.0971
    0.9340    0.3922    0.0318    0.8235

```

4. ones: 产生 “1” 矩阵或数组

其调用格式如下:

A=ones(n): 生成 $n \times n$ 的全 1 矩阵。

A=ones(m, n): 生成 $m \times n$ 的全 1 阵。如果 $m=1$, 则生成行向量; 若 $n=1$, 则生成列向量。此种调用的另一种方式是: **I=ones([m, n])**, 其效果和 **I=ones(m, n)** 一样。

A=ones(d1, d2, d3,...): 生成 $d1 \times d2 \times d3 \times \dots$ 的全 1 多维数组。

A=ones(size(B)): 生成与矩阵 A 行列数相同的全 1 阵。

说明: 矩阵或数组的所有元素为 1。其余语法格式与 zeros 函数与 eye 函数类似。

【例 1-13】ones 函数用法示例。

```

>> x = ones(2,3,'int8')

x =

     1     1     1
     1     1     1

>> x = ones(3,3)

x =

     1     1     1

```

```
1    1    1
1    1    1
```

如果文件“exam.m”保存在搜索路径中，在 MATLAB 命令窗口中输入 exam 可产生矩阵 A，这对经常输入大矩阵或需要输入多个矩阵时，相当方便。

利用外部数据文件装入到指定矩阵，可以通过 MATLAB 提供的文件输入、输出函数来实现。

1.6 矩阵的运算

矩阵运算是 MATLAB 的核心。虽然在 MATLAB 中，处理的数据类型是多维数组，但通常大多数时候所处理的数据，都是矩阵形式的。

1.6.1 矩阵的代数运算

1. 加减法运算

在 MATLAB 中，对矩阵进行加法运算、减法运算与在线性代数中进行的矩阵运算一样。其执行格式如下：

$C=A+B$ 或 $C=plus(A,B)$

$C=A-B$ 或 $C=minus(A,B)$

对矩阵 A 和矩阵 B 进行加法或者减法运算，必须要求矩阵 A 和矩阵 B 的大小一样，即矩阵 A 和矩阵 B 具有相同的行数和列数；否则，MATLAB 将给出出错信息。在以下的内容中，如果说两个矩阵大小一样，是指两个矩阵的行数和列数相同。

特殊的情况，如果矩阵 A 或者矩阵 B 中有一个是标量，MATLAB 允许标量和任意大小的矩阵进行加减运算，运算的结果是把矩阵的每一个元素都和这个标量进行加减。

在 MATLAB 中，算术运算符“+”和“-”也可以作为一元运算符使用。+A 就是取矩阵 A，而-A 则是对矩阵 A 中的每个元素取相反数。

【例 1-14】矩阵的加减法运算。

```
>> A=[1 3 7;2 4 6];
>> B=[11 7 9;8 6 3];
>> C=A+B
C =
    12    10    16
    10    10     9
>> D=A-B
D =
   -10    -4    -2
    -6    -2     3
>> E=A+2
E =
     3     5     9
     4     6     8
```

2. 矩阵的乘法运算

在 MATLAB 中，对矩阵进行乘法运算可分为按矩阵的乘法运算和按数值的乘法运算两种。这两种运算的结果是不一样的。其调用格式如下：

矩阵乘法: $C=A*B$ 或 $C=mtimes(A,B)$;

数值乘法: $C=A.*B$ 或 $C=times(A, B)$;

矩阵乘法 $C=A*B$ 就是通常代数学中的矩阵乘法, 要求矩阵 A 的列数必须等于矩阵 B 的行数, 如果不满足这一条件, MATLAB 会显示出错信息。

数值乘法 $C=A.*B$, 运算符用的是点乘 “.”, 表示矩阵 A 和矩阵 B 对应的元素分别相乘, 要求矩阵 A 和矩阵 B 的大小一样, 就是说, 矩阵 A 和矩阵 B 的行数与列数必须相等。

标量可以与任意大小的矩阵相乘, 运算结果是把矩阵的每一个元素都乘以这个标量。这也就是通常代数学中的数值与矩阵的乘法。

【例 1-15】 矩阵的乘法运算示例。

```
>> A=[1 3 7;2 4 6];
>> B=[11 7 9;8 6 3];
>> C=[1 2;3 4;5 6];
>> D=A*C
D =
    45    56
    44    56
>> E=A.*B
E =
    11    21    63
    16    24    18
>> F=A*B
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>> J=A*4
J =
     4    12    28
     8    16    24
>> K=A.*4
K =
     4    12    28
     8    16    24
```

从例子中可以看出, 矩阵和一个标量, 进行矩阵乘或是数值相乘, 其结果是一样的。

3. 矩阵除法运算

1) 矩阵求逆

行数和列数相等的矩阵称为方阵, 在代数学中如果方阵的行列式不等于零, 则称方阵是可逆的, 可逆方程有逆矩阵。在 MATLAB 中, 求方阵的逆矩阵的函数是: $B=inv(A)$ 。

【例 1-16】 矩阵求逆示例。

```
>> n = 500;
Q = orth(randn(n,n));
d = logspace(0,-10,n);
A = Q*diag(d)*Q';
x = randn(n,1);
b = A*x;
tic,
y = inv(A)*b
```

```

toc
y =
    -0.8583
     1.4998
     .....
    -0.4883
     1.3104
Elapsed time is 0.261187 seconds.

```

2) 除法运算

在线性代数中，只有逆矩阵的定义，而没有矩阵除法的运算。而在 MATLAB 中，特别定义了矩阵除法运算。矩阵除法运算在 MATLAB 中是一个十分有用的运算。根据实际问题的需要，定义了两种除法命令：左除和右除。而矩阵的左除和右除运算，又分别有按矩阵除法运算和按数值除法运算。

(1) 矩阵除法运算。矩阵除法的运算调用格式如下：

矩阵左除：C=A\B 或 C=mldivide(A, B)

矩阵右除：C=A/B 或 C=mrdivide(A, B)

通常矩阵左除不等于右除，如果 A 是可逆方阵，A\B 表示方程 AX=B 的解，其等效于 A 的逆矩阵左乘矩阵 B，也就是 inv(A)*B；B/A 表示方程 XA=B 的解，其等效于 B*inv(A)（即 B 右乘 A 的逆矩阵）。

【例 1-17】解矩阵方程。

$$\textcircled{1} \begin{bmatrix} 2 & 5 \\ 1 & 3 \end{bmatrix} X = \begin{bmatrix} 4 & -6 \\ 2 & 1 \end{bmatrix}; \quad \textcircled{2} Y \begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 0 & -1 \end{bmatrix}$$

其实现的 MATLAB 程序代码如下：

```

>> A=[2,5;1,3]; B=[4,-6;2,1];
C=[2,0;-1,1]; D=[3,1;0,-1];
X=A\B
Y=D/C

```

运行程序，输出如下：

```

X =
     2    -23
     0     8
Y =
     2.0000     1.0000
    -0.5000    -1.0000

```

求出的结果 X=A\B 就是矩阵方程 (1) 的解；结果 Y=D/C 就是矩阵方程 (2) 的解。

(2) 数值除法运算。数值除法的运算调用格式如下：

数值左除：C=A.\B 或 C=ldivide(A, B)

数值右除：C=A./B 或 C=rdivide(A, B)

对矩阵进行数值除法，是矩阵中对应元素进行除法运算，这要求矩阵 A 和矩阵 B 大小相同，即矩阵 A 和矩阵 B 的行数和列数是一样的。数值左除 A.\B 是矩阵 A 和 B 中对应元素相除，即 B(i,j)/A(i,j)；数值右除 A./B 是矩阵 B 和矩阵 A 中对应元素相除，即 A(i, j)/B(i,j)。由此可知：A./B=B.\A。要注意矩阵的数值除法的运算符是点除。

【例 1-18】数值除法示例。

```
>> A=[1 2 3;4 5 6;7 8 9];
B=[1 3 7;2 5 8;3 6 9];
C=A./B
D=A.\B
```

运行程序，输出如下：

```
C =
    1.0000    0.6667    0.4286
    2.0000    1.0000    0.7500
    2.3333    1.3333    1.0000
D =
    1.0000    1.5000    2.3333
    0.5000    1.0000    1.3333
    0.4286    0.7500    1.0000
```

4. 矩阵的乘方运算

在 MATLAB 中，乘方运算也分为矩阵乘方运算和数值乘方运算两种。

矩阵乘方运算： $C=A^B$ 或 $C=\text{mpower}(A, B)$

数值乘方运算： $C=A.^B$ 或 $C=\text{power}(A, B)$

矩阵乘方运算要求 A 是一个方阵，且 B 是一个标量， A^B 的意思是矩阵 A 的 B 次方。如果 B 是个正整数，则 A^B 是矩阵 A 自乘 B 次。例如， $A^3=A*A*A$ 。如果 B 是一个负整数，则首先对 A 求逆，然后将它自乘 B 次。例如， $A^{-3}=(\text{inv}(A))*(\text{inv}(A))*(\text{inv}(A))$ 。如果 B 不是整数，则涉及特征值和特征向量的求解问题。例如，若已知矩阵 A 的特征值矩阵为 D ，特征向量矩阵为 V ，则有 $A^B=V*(D.^B)/V$ ，其中 D 是对角阵， $D.^B$ 为数值的乘方。

数值的乘方 $A.^B$ 就是矩阵 A 和 B 对应元素的乘方，即 $A(i, j)$ 和 $B(i, j)$ 次方。除非其中一个为标量，否则矩阵 A 和矩阵 B 的行列数必须相等。如果 A 是标量， B 是矩阵，结果 $A.^B$ 是一个矩阵，其元素是 A 的 $B(i, j)$ 次方；如果 B 是标量， A 是矩阵，则 $A.^B$ 是 A 中的每个元素 $A(i, j)$ 的 B 次方。

【例 1-19】矩阵的乘方运算示例。

```
>> A=[4 0 0;0 3 1;1 2 4];
B=A^3
B =
    64     0     0
    11    47    39
    50    78    86
>> C=A^0.2
C =
    1.3195     0     0
   -0.0084    1.2257    0.0770
    0.0686    0.1540    1.3027
>> D=A.^(-3)
D =
    0.0156     Inf     Inf
     Inf    0.0370    1.0000
    1.0000    0.1250    0.0156
```

5. 矩阵的转置与共轭转置

在 MATLAB 中，对复数矩阵可以进行转置和共轭转置两种运算。

复矩阵的共轭转置： $B=A'$ 或 $B=\text{ctranspose}(A)$

复矩阵的转置： $B=A.'$ 或 $B=\text{transpose}(A)$

要注意，转置运算符是点运算，即“.”。

【例 1-20】矩阵的转置与共轭转置示例。

```
>> A=eye(2,3)+i*ones(2,3)
A =
    1.0000 + 1.0000i    0 + 1.0000i    0 + 1.0000i
    0 + 1.0000i    1.0000 + 1.0000i    0 + 1.0000i
>> B=A',C=ctranspose(A)
B =
    1.0000 - 1.0000i    0 - 1.0000i
    0 - 1.0000i    1.0000 - 1.0000i
    0 - 1.0000i    0 - 1.0000i
C =
    1.0000 - 1.0000i    0 - 1.0000i
    0 - 1.0000i    1.0000 - 1.0000i
    0 - 1.0000i    0 - 1.0000i
>> D=A.',E=transpose(A)
D =
    1.0000 + 1.0000i    0 + 1.0000i
    0 + 1.0000i    1.0000 + 1.0000i
    0 + 1.0000i    0 + 1.0000i
E =
    1.0000 + 1.0000i    0 + 1.0000i
    0 + 1.0000i    1.0000 + 1.0000i
    0 + 1.0000i    0 + 1.0000i
```

从运行的结果可以看出，转置运算符和转置运算函数的运算结果一样，并且只是对矩阵进行转置，没有进行共轭转置。

如果矩阵 A 是实矩阵，则对矩阵进行转置或共轭转置，运算结果是一样的。

例如：

```
>> A=[4,7,2,9;9,1,4,3;1,9,6,4]
B=A', C=A.'
A =
     4     7     2     9
     9     1     4     3
     1     9     6     4
B =
     4     9     1
     7     1     9
     2     4     6
     9     3     4
C =
     4     9     1
     7     1     9
```


2 4 6
9 3 4

需要注意的是，转置和共轭转置只能对矩阵运算，不能对多维数组运算。

6. 矩阵的函数运算

1) 常用数学函数

MATLAB 函数大部分都适于做数组运算，只有专门说明的几个除外，如*、/、\、运算符和指数函数 expm、对数函数 logm、开方函数 sqrtm。表 1-1 所列基本函数库中的常用函数都可用于数组运算，即其自变量都可以是任意阶的矩阵。

表 1-1 基本函数库 (elfun) (未标注输入变元的为单输入/输出函数)

三角函数	sin	正弦	asin	反正弦	coth	双曲余切
	cos	余弦	acos	反余弦	acoth	反双曲余切
	tan	正切	atan	反正切	sech	双曲正割
	cot	余切	acot	反余切	asech	反双曲正割
	sec	正割	asec	反正割	csch	双曲余割
	csc	余割	acsc	反余割	acsch	反双曲余割
指数函数	sinh	双曲正弦	asinh	反双曲正弦	atan2(x,y)	4 象限反正切
	cosh	双曲余弦	acosh	反双曲余弦		
	tanh	双曲正切	atanh	反双曲正切		
	exp	以 e 为底的指数	log	自然对数	nextpow2	取最接近较大的 2 次幂
	log2	以 2 为底的指数	pow2	2 的幂		
	log10	以 10 为底的指数	sqrt	方根		
复数	abs	绝对值和复数模值	angle	相角	unwrap	去掉相角突变
	real	实部	conj	共轭复数	cplxpair	按复数共轭对排序元素群
	imag	虚部	isreal	是实数时为真		
取整函数	round	四舍五入为整数	floor	向 $-\infty$ 舍入为整数	rem(a,b)	a 整数 b, 求余数
	fix	向 0 舍入为整数	sign	符号函数	mod(x,m)	x 整数 m 取正余数
	ceil	向 ∞ 舍入为整数				

下面举例说明数组运算的优越性。

【例 1-21】要求列出一个三角函数表。

```
>> x=[0:0.1:pi/4]';
>> [x,sin(x),cos(x),tan(x)]
```

第一条语句把数组 x 赋值，经转置后成为一个列向量。因为 sin、cos、tan 函数都对元素群有效，得出的都是同阶的列向量。第二条语句把 4 个列向量组成一个矩阵并显示，得

```
ans =
    0         0    1.0000         0
  0.1000    0.0998    0.9950    0.1003
  0.2000    0.1987    0.9801    0.2027
  0.3000    0.2955    0.9553    0.3093
  0.4000    0.3894    0.9211    0.4228
  0.5000    0.4794    0.8776    0.5463
```

0.6000	0.5646	0.8253	0.6841
0.7000	0.6442	0.7648	0.8423

第一列是 x ，以下各列依次是 $\sin(x)$ 、 $\cos(x)$ 、 $\tan(x)$ 。

2) 数据分析函数

数据分析函数是按列运算的，见表 1-2。

表 1-2 常用的数据分析函数

函 数 名	功 能	函 数 名	功 能
max	最大元素	min	最小元素
mean	均值	sum	求和
prod	求积		

【例 1-22】数据分析函数示例。

```
>> A=[4,7,2,9;9,1,4,3;1,9,6,4]
```

```
a=max(A)
```

```
b=sum(A)
```

```
c=prod(A)
```

运行程序，输出如下：

```
A =
```

```
4     7     2     9
```

```
9     1     4     3
```

```
1     9     6     4
```

```
a =
```

```
9     9     6     9
```

```
b =
```

```
14    17    12    16
```

1.6.2 矩阵关系运算

MATLAB 也支持关系运算。和许多编程语言类似，MATLAB 把非零数当作“真”，把零当作“假”。对于所有的关系表达式，MATLAB 把“真”值输出为“1”；而把“假”值输出为“0”。

MATLAB 关系运算符有：小于（<）、小于等于（<=）、大于（>）、大于等于（>=）、相等（==）、不相等（~=）。

在下面的关系运算中，A 和 B 是行、列数相等的矩阵，也可以 A、B 中有一个是标量，另一个是矩阵，比如 A 是矩阵，B 是标量，运算时把矩阵 A 中的每一个元素都与标量 B 比较。

1. 小于

小于的关系表达式如下：

$C = (A < B)$ 或 $C = A < B$ ：如果矩阵 A 中的元素小于矩阵 B 中对应位置的元素，则在结果矩阵 C 中输出“1”，否则输出“0”。

小于关系的另一种表达方式是用函数“ $C = \text{lt}(A, B)$ ”。

2. 大于

大于的关系表达式如下：

$C = (A > B)$ 或 $C = A > B$ ：如果矩阵 A 中的元素大于矩阵 B 中对应位置的元素，则在结果矩阵 C 中输出“1”，否则输出“0”。

大于关系的另一种表达方式是用函数 “ $C=gt(A, B)$ ”。

3. 小于等于

小于等于的关系表达式如下：

$C=(A<=B)$ 或 $C=A<=B$ ：如果矩阵 A 中的元素小于或等于矩阵 B 中对应位置的元素，则在结果矩阵 C 中输出 “1”，否则，输出 “0”。

小于等于关系的另一种表达方式是用函数 “ $C=le(A, B)$ ”。

4. 大于等于

大于等于的关系表达式如下：

$C=(A>=B)$ 或 $C=A>=B$ ：如果矩阵 A 中的元素大于或等于矩阵 B 中对应位置的元素，则在结果矩阵 C 中输出 “1”，否则，输出 “0”。

大于等于关系的另一种表达方式是用函数 “ $C=ge(A, B)$ ”。

5. 等于

相等的关系表达式如下：

$C=(A==B)$ 或 $C=A==B$ ：如果矩阵 A 中的元素与矩阵 B 中对应位置的元素相等，则在结果矩阵 C 中输出 “1”，否则输出 “0”。

相等关系的另一种表达方式是用函数 “ $C=eq(A, B)$ ”。

6. 不等

不等关系的表达式如下：

$C=(A\sim=B)$ 或 $C=A\sim=B$ ：如果矩阵 A 中的元素与矩阵 B 中对应位置的元素不相等，则在结果矩阵 C 中输出 “1”，否则输出 “0”。

【例 1-23】矩阵关系运算示例。

```
>> A=[1 2 3;4 5 6]; B=[1 7 9;0 5 4];
C=A>B
D=gt(A,B)
E=A==B
F=(A==B)
```

运行程序，输出如下：

```
C =
    0     0     0
    1     0     1

D =
    0     0     0
    1     0     1

E =
    1     0     0
    0     1     0

F =
    1     0     0
    0     1     0
```

从运算的结果可以看出，用 “ $C=A==B$ ” 和 “ $D=(A==B)$ ” 的结果相同。

关系运算在编程时经常用于分支结构或循环结构。按照 MATLAB 操作符的优先级，表达式 $C=A==B$ ，先判断 $A==B$ ，然后赋值给矩阵 C ，但在对程序进行阅读时，却很容易搞错，因此最好把表达式改写为 $D=(A==B)$ ，使得更容易阅读。对于其他形式的关系表达式，也采用

类似的表示形式为好。

1.6.3 矩阵的逻辑运算

MATLAB 中除了提供矩阵的算术运算和关系运算外，还提供了矩阵的逻辑运算。

MATLAB 中有四种逻辑运算符： $\&$ （与）、 \mid （或）、 \sim （非）、 xor （异或）。在 MATLAB 中，逻辑运算的结果输出时，用“1”代表真，用“0”代表“假”。但是在判断一个量是否为真时，以任意的非零数代表“真”，以“0”代表“假”。另外，逻辑运算符按元素进行比较，参与逻辑运算的两个对象可以都是矩阵，也可以都是标量，还可以其中之一是矩阵，另一个是标量。当其中之一为标量时，将该标量与另一矩阵的所有元素进行逻辑运算，返回结果仍是由“0”和“1”组成的矩阵，并且该矩阵和参加运算的矩阵具有相同的行数和列数。

1) 与运算

逻辑“与”运算的结果是一个矩阵，当参加运算的两个元素都是非零数时，结果为“1”；当参加运算的两个元素至少有一个为“0”时，结果为“0”。逻辑“与”运算的格式如下：

$C = (A \& B)$ 或者 $C = A \& B$ 或者 $C = \text{and}(A, B)$

2) 或运算

逻辑“或”运算的结果是一个矩阵，当参加运算的两个元素有一个是非零数时，结果为“1”；当参加运算的两个元素都为“0”时，结果为“0”。逻辑“或”运算的格式如下：

$C = (A \mid B)$ 或者 $C = A \mid B$ 或者 $C = \text{or}(A, B)$

3) 非运算

逻辑“非”运算的结果是一个矩阵，当参加运算的矩阵中的元素是非零数时，结果为“0”；当参加运算元素为“0”时，结果为“1”。逻辑“非”运算的格式如下：

$C = (\sim A)$ 或者 $C = \sim A$ 或者 $C = \text{not}(A)$

4) 异或

逻辑“异或”运算的结果是一个矩阵，当参考运算的两个元素有一个为“0”、另一个是非零数时，结果为“1”；当参加运算的两个元素都为“0”或者都为非零数时，结果为“0”。逻辑“异或”运算的格式如下：

$C = \text{xor}(A, B)$

5) 快速运算

在进行逻辑与、逻辑或运算时，还可以用快速运算符。

(1) 快速逻辑与运算。快速逻辑与运算的格式如下：

$C = (A \&\& B)$ 或 $C = A \&\& B$

快速与运算的规则是：如果 A 的值为 0，则直接赋值 C 为 0，不运算 B。如果 A 的值为 1，再计算 B 的值。

因为如果 A 的值为 0，那么 A 和 B 进行与运算的结果应该为 0。

使用快速与运算，可以防止程序中一些不可预知的错误，例如，用语句：“ $x = (b \sim = 0) \&\& (a/b > 19)$ ”，只有当 b 不为 0 时，才进行除法运算。

(2) 快速逻辑或运算。快速逻辑或运算的格式如下：

$C = (A \mid\mid B)$ 或 $C = A \mid\mid B$

快速或运算的规则是：如果 A 的值为 1，则直接赋值 C 为 1，不运算 B。如果 A 的值为 0，再计算 B 的值。

【例 1-24】矩阵的逻辑运算示例。

```
>> A=[1 2;0 4]; B=eye(2);
C1=(A|B), C2=A|B
D1=(A&B), D2=A&B
E=xor(A,B)
F=not(A)
```

运行程序，输出如下：

```
C1 =
     1     1
     0     1
C2 =
     1     1
     0     1
D1 =
     1     0
     0     1
D2 =
     1     0
     0     1
E =
     0     1
     0     0
F =
     0     0
     1     0
```


1.7 MATLAB 帮助系统

有效地使用帮助系统中所提供的信息，是用户掌握好 MATLAB 应用的最佳途径。熟练的程序开发人员总会充分地利用软件提供的帮助信息，而 MATLAB 的一个突出优点就是其拥有较为完善的帮助系统。MATLAB 的帮助系统可以分为联机帮助系统和命令窗口查询帮助系统。本节将对这两种帮助系统分别进行介绍。

1.7.1 联机帮助系统

1. 进入联机帮助系统

MATLAB 的联机帮助系统非常全面，几乎包括该软件的所有内容。可以选择如下 3 种方式进入 MATLAB 的帮助系统。

- (1) 在 MATLAB 主窗口中，单击【Help】菜单下的【Product Help】命令。
- (2) 按“F1”快捷键，系统将弹出帮助窗口。
- (3) 按主窗口工具栏中的“（帮助）”按钮，进入帮助窗口。

2. 帮助导向界面

在联机帮助系统中，左侧部分为帮助导向界面，右侧为帮助显示界面。

帮助导向界面下侧的 4 个选项分别为 Contents（帮助主题）、Index（帮助索引）、Search Results（帮助查询）和 Demos（联机演示）。下面分别介绍它们使用方法。

1) Contents (帮助主题)

单击该选项卡，将提示 MATLAB 的帮助内容，如图 1-8 所示。在 MATLAB 弹出的帮助窗口中，左侧相当于一个目录系统，列出了 MATLAB 帮助系统中所包含的各项主要内容，如图 1-8 所示。单击其中的任意一项，窗口的右侧将显示该项内容的具体解释。初始状态时，右侧显示的内容是关于帮助系统的一些介绍，指导用户对帮助系统进行充分的运用。

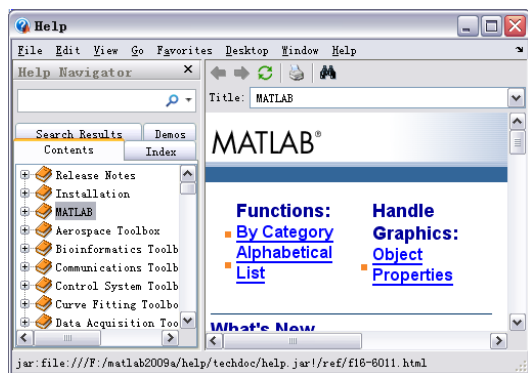


图 1-8 联机帮助系统

2) Index (帮助索引)

单击该选项卡，在“Enter index term”文本框中输入用户需要查找的内容，右边的窗口中会显示关于该内容的相关信息。如用户要学习 loglog 函数的用法，可以在“Enter index term”文本框中输入“loglog”命令，按“Enter”键，右侧窗口中立即显示出相关的信息，如语法、描述和与其相关的函数等。

3) Search (帮助查询)

单击该 Search 选项卡，在“Search for”下拉列表框中选择或输入文件名。如在本例中输入函数名“loglog”，单击“Go”按钮，就会在右侧窗口中输入相关文件和 loglog 的相关信息，如图 1-9 所示。

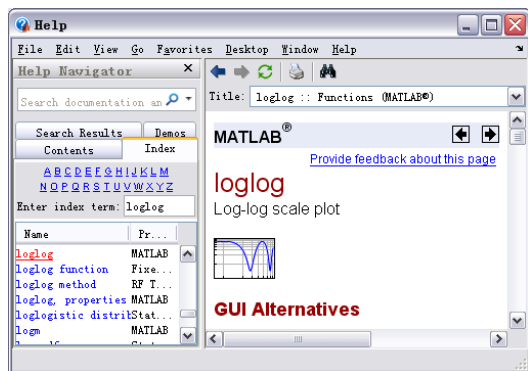


图 1-9 查询 loglog 函数显示页面

注意，虽然使用帮助索引和帮助查询都可以得知某个具体函数的使用方法，但是两者还是有区别的。在使用帮助索引时，左侧显示的信息是按字母排序的，这些信息很多与 loglog 函数并无关系。而使用帮助查询时，右侧显示的是与 loglog 函数相关的一些信息。使用这些信息比使用 index 所查询的信息要丰富很多。

4) Demos (联机演示)

MATLAB 除了常规的帮助系统外，还设立了联机演示系统，对于初学者来说，查看 MATLAB 的联机演示是最佳的学习方法。在该项内容中，MATLAB 设置了许多关于各个工具箱内容的现成程序，用户可以选择自己所需的部分来学习相关内容。

用户可以使用如下两种方法进行联机演示，其效果如图 1-10 所示。

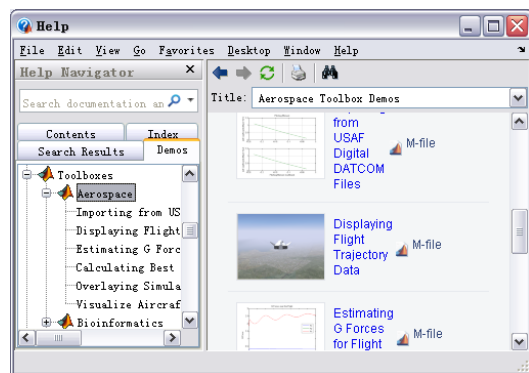


图 1-10 联机演示页面

(1) 单击帮助窗口中的“Demos”按钮。此时联机演示窗口包含两个部分，左边的部分是项目栏，用户可以来选所需演示的项目；右边是对此项目的说明文字。双击左边项目栏的具体内容，或单击该内容，再单击右边说明框中的“Run this demo”或“Run in the Command Window”或“Open this model”超链接，MATLAB 都将会弹出新的窗口进行联机演示操作。

(2) 在命令窗口中运行“Demos”函数，也可以进入 MATLAB 的联机演示界面。

1.7.2 命令窗口查询帮助系统

当用户对 MATLAB 有一定了解后，可以通过在命令窗口中直接输入命令来获得相关的帮助信息，这种获取方式比联机帮助更为快捷、直接。在命令窗口中获取帮助信息的主要命令为 help 和 lookfor 函数。

1. help 函数

help 函数有 4 种用法，分别是 help、help+函数名（函数类名）、helpdesk、helpwin。

下面分别介绍 help 和 help+函数名（函数类名）这两种用法。

1) help 命令

在命令窗口中直接输入“help”命令，会显示当前帮助系统中所包含的所有项目。需要注意的是，用户在输入该命令后，命令窗口只显示当前搜索路径中的所有目录名称。

```
>> help
HELP topics:
My Documents\MATLAB
matlab\general
matlab\ops
matlab\lang
matlab\elmat
matlab\randfun
matlab\elfun
.....
- (No table of contents file)
- General purpose commands.
- Operators and special characters.
- Programming language constructs.
- Elementary matrices and matrix manipulation.
- Random matrices and random streams.
- Elementary math functions.
.....
```

xpc\xpc	- xPC Target
xpcblocks\thirdpartydrivers	- (No table of contents file)
build\xpcblocks	- (No table of contents file)
xpc\xpcdemos	- xPC Target -- demos and sample script files.
kernel\embedded	- xPC Target Embedded Option

2) help+函数名 (函数类名)

当用户知道某个函数名称, 如果要了解该函数的具体用法, 只需在命令窗口中输入“help+函数名”。如果用户想了解 **diag** 函数的具体用法, 只需在命令窗口中输入“help diag”, 即可得到如下的关于此函数的基本信息。

```
>> help diag
DIAG Diagonal matrices and diagonals of a matrix.
    DIAG(V,K) when V is a vector with N components is a square matrix
    of order N+ABS(K) with the elements of V on the K-th diagonal. K = 0
    is the main diagonal, K > 0 is above the main diagonal and K < 0
    is below the main diagonal.
    DIAG(V) is the same as DIAG(V,0) and puts V on the main diagonal.
    DIAG(X,K) when X is a matrix is a column vector formed from
    the elements of the K-th diagonal of X.
    DIAG(X) is the main diagonal of X. DIAG(DIAG(X)) is a diagonal matrix.
Example
    m = 5;
    diag(-m:m) + diag(ones(2*m,1),1) + diag(ones(2*m,1),-1)
produces a tridiagonal matrix of order 2*m+1.
See also spdiags, triu, tril, blkdiag.
Overloaded methods:
    codistributed/diag
    gf/diag
    frd/diag
    uss/diag
    umat/diag
    ultidyn/diag
    ufrd/diag
    ndlft/diag
    frd/diag
    sym/diag
Reference page in Help browser
doc diag
```

同样, 当用户要知道某一函数类型的具体用法, 只需在命令窗口中输入“help+函数类名”即可。如用户要得到 **matfun** 函数类型的具体用法, 只需在命令窗口中输入“help matfun”, 即可得到如下的关于此函数的类型的基本信息。

```
>> help matfun
Matrix functions - numerical linear algebra.
Matrix analysis.
    norm          - Matrix or vector norm.
    normest       - Estimate the matrix 2-norm.
    rank          - Matrix rank.
```


det - Determinant.
 trace - Sum of diagonal elements.
 null - Null space.
 orth - Orthogonalization.
 rref - Reduced row echelon form.
 subspace - Angle between two subspaces.

Linear equations.

/ and / - Linear equation solution; use "help slash".

Factorization utilities

qrdelete - Delete a column or row from QR factorization.
 qrinsert - Insert a column or row into QR factorization.
 rsf2csf - Real block diagonal form to complex diagonal form.
 cdf2rdf - Complex diagonal form to real block diagonal form.
 balance - Diagonal scaling to improve eigenvalue accuracy.
 planerot - Givens plane rotation.

2. lookfor 函数

一般来说,当用户知道某个函数的具体名称时,可以使用 `help` 函数寻找到相关的帮助信息。但是对于初学者来说,往往不知道函数的确切名称,在这种情况下使用 `lookfor` 函数可以很方便地解决这个问题。在使用 `lookfor` 函数时,用户只需知道某个函数的部分关键字,在命令窗口中输入“`lookfor+关键字`”,就可以很方便地实现查找。如用户需要查找含有关键字 `norm` 的相关内容,即可按照下述的方法来实现。

```
>> lookfor norm
cgnormaliser - cgnormfunction - Constructor for the
cgnormfunction class.
cgnormnode - construct a cgnormnode object
realmin - Smallest positive normalized floating point number.
randn - Normally distributed pseudorandom numbers.
condest - 1-norm condition number estimate.
norm - Matrix or vector norm.
normest - Estimate the matrix 2-norm.
.....
normspec - Plots normal density between specification limits.
normstat - Mean and variance for the normal distribution.
wgtnormfit - Fitting demo for a weighted normal distribution.
wgtnormfit2 - Fitting demo for a weighted normal distribution
(log(sigma) parameterization).
Remove2DFFTWarnForNormalizeCB - Remove the warning generated from the block
```

此外,还有 `doc`、`docsearch`、`helpbrowser`、`matlabpath`、`more`、`partialpath`、`which`、`whos`、`who` 和 `class` 等函数与此相关,用户可以学以致用,自己使用帮助系统查看这些函数的使用方法和功能。

第 2 章 MATLAB 的程序设计及数值计算

2.1 MATLAB 程序结构

程序的结构有顺序结构、分支结构和循环结构三种。任何复杂的程序都是由这三种基本结构所构成的。

2.1.1 顺序结构

顺序结构是指按照程序中语句的排列顺序依次执行，直到程序的最后一个语句。这是最简单的一种程序结构。一般涉及数据的输入、输出及程序的暂停。

1. 数据的输入

从键盘上输入数据，可以使用 `input` 函数，其调用格式如下：

`a=input` (提示信息, 选项): 其中，提示信息为一个字符串，用于提示用户输入什么样的数据。

例如，从键盘输入正整数 `n`，可以采用以下命令来完成：

```
n=input('输入正整数 n=');
```

执行该语句时，首先在屏幕上显示提示信息“输入正整数 `n=`”，然后等待用户从键盘上输入正整数 `n` 的值。

如果在 `input` 函数调用时采用 `'s'` 选项，则允许用户输入一个字符串。例如，想输入一个人的姓名，可采用命令：

```
name=input('请输入姓名','s')
```

2. 数据的输出

MATLAB 提供的命令窗口输出函数主要有 `disp`、`fprintf` 函数。`disp` 函数调用格式如下：

`disp`(输出项): 输出项可以是字符串，也可以是矩阵。例如：

```
>> A='您好';
```

```
>> disp(A)
```

输出如下：

```
您好
```

又如：

```
>> A=[1 4 7;2 5 8;3 6 9];
```

```
>> disp(A)
```

输出如下：

```
1     4     7
2     5     8
3     6     9
```

`fprintf` 函数最常见的使用方式用以下例子来说明。

若输入命令：

```
fprintf('圆周率 pi=%10.9f',pi)
```

则会按浮点型数输出含 9 位小数、1 位整数的圆周率近似值，其输出结果如下：

```
圆周率 pi=3.141592654
```

若键入命令：

```
>> n=24; fprintf('n=%d',n)
```

则会按整型数输出 n 的值，其输出结果如下：

```
n=24
```

若输入命令：

```
>> n=24; fprintf('n=%f',n)
```

则会按浮点型数输出 n 的值，其输出结果如下：

```
n=24.000000
```

3. 程序的暂停

当程序运行时，为了查看程序的中间结果或者观看输出的图形，有时需要暂停程序的执行，这时可以使用 pause 函数，其调用格式如下：

```
pause(延迟秒数)
```

如是省略延迟秒数，则将暂停程序，直到用户按任一键后程序继续执行。

2.1.2 分支结构

在复杂的计算中，常常需要根据表达式的情况（是否满足某些条件）确定下一步该做什么。MATLAB 的 if-else-elseif 与 switch 语句提供了描述条件分支的结构。

1. if-else-elseif 语句

if 语句用来检查逻辑运算、逻辑函数、逻辑变量值等逻辑表达式的真假，若为真则执行 if 和 else 之间的执行语句；否则，转去执行另一分支。其语法格式如下：

```
if 逻辑表达式
```

```
    执行语句 1
```

```
else
```

```
    执行语句 2
```

```
end
```

在 MATLAB 中也可以利用 elseif 来写嵌套判断式，其语法格式如下：

```
if 逻辑表达式 1
```

```
    执行语句 1
```

```
elseif 逻辑表达式 2
```

```
    执行语句 2
```

```
elseif 逻辑表达式 3
```

```
    执行语句 3
```

```
else
```

```
    执行语句 4
```

```
end
```

【例 2-1】利用 MATLAB 的语句实现多路选择。

(1) 编写函数文件 bspline.m 计算 B 样条曲线。

```
function f = bspline(x)
```

```
if x<0
```

```

    f = 0;
elseif x<1
    f = x;
elseif x<2
    f = 2-x;
else
    f = 0;
end

```

(2) 利用函数文件，进行实数分区间选择。

```

>>bspline(-1)
ans =
    0
>>bspline(1.6)
ans =
    0.4000
>>bspline(2)
ans =
    0

```

【例 2-2】列出 88~188 以内的所有素数，并求出它们的和。

其实现的 MATLAB 程序代码如下：

```

>> clear all;
sum=0; ss=[];
for i=88:188,
    j=0;
    for k=1:i
        if(rem(i,k)==0) %求 i 除以 k 的余数
            j=j+1;
        end
    end
    if(j<=2)
        ss=[ss,i];
        sum=sum+i;
    end
end
disp('88 至 188 以内的素数是: ');
disp(ss);
disp('88 至 188 以内的素数之和为: ');
disp(sum);

```

运行程序，输出如下：

88 至 188 以内的素数是：

Columns 1 through 12

89 97 101 103 107 109 113 127 131 137 139 149

Columns 13 through 19

151 157 163 167 173 179 181

88 至 188 以内的素数之和为：

2573

2. switch 语句

switch 语句也是 MATLAB 中的一个分支语句。如果在一个程序中，必须针对某个变量值来进行多种不同的执行，switch 语句更为方便，此外，合理地使用 switch 语句也可以使程序更具有可读性。

switch 的语法结构如下：

```
case 数值（或字符串）条件语句 1
    执行语句 1
case 数值（或字符串）条件语句 2
    执行语句 2
otherwise
    执行语句 n
end
```

基本的 switch 语句包含下列元素：

- (1) **switch**: switch 语句的开始，紧接着分支条件。分支条件可以是一个函数或表达式。
- (2) **case**: 依照分支条值，不同 case 可以定义为不同的运算指令，而紧接在 case 后面的就是此 case 的分支条件。之后接着一个或一串执行语句。
- (3) **otherwise**: 若不符合所有 case 的条件，则程序就会执行 otherwise 下面的语句。
- (4) **end**: switch 语句的结束。

switch 实际上也是利用 if（分支条件 == 数值条件）比较来实现其目的，如果判断为真，则符合条件，并且执行紧跟的运算；反之，如判断的返回值为假，则继续检验 case，直到最后一个失败，才执行 otherwise 下面的程序片段。如果分支条件是用于检测字符串条件，则其判断变为

if (strcmp (分支条件 == 字符串条件))

switch 这一结构也经常用于一般的程序语言中，比如 C 语言。但如果读者学过 C 语言，也许会有这样的疑问，即 MATLAB 的 switch 语句中为什么缺少了 break。其实很简单，在 C 语言中，程序执行所有的符合条件的 case。换句话说，C 语言在检验某个 case 符合并执行其后的语句后，还会再继续检验下一个 case，直到全部检验完。所以，一般会在一个 case 描述的最后加上 break，让程序只运算第一个检验成功的运算式。MATLAB 在这方面则只执行第一个检验成功的 case。

【例 2-3】试绘制二阶系统 $y = 1 + \frac{e^{-\xi\omega_n t}}{\beta} \sin(\omega_n \beta t + \theta)$ 的单位阶跃响应图形，其中 $\beta = \sqrt{1 - \xi^2}$ ，

$\theta = \arctan \sqrt{\frac{1 - \xi^2}{\xi}}$ ， $\omega_n = 0.2$ ， $\xi = 0.25、0.5、0.75$ ，时间范围为 1~80s。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=0:0.1:80;      %定义时间向量
for ksi=0.3:0.3:0.9
    be=sqrt(1-ksi^2);
    bi=atan(be/ksi);
    wn=0.2;
```

```

y=1-(exp(-ksi*wn.*t)./be).*sin((be*wn).*t+bi);
switch round(ksi*10) %根据不同的 ke,曲线采用不同的颜色
    case 3
        plot(t,y,'r:'); %当 ke=0.25,用黑色点连线
        hold on
    case 6
        plot(t,y,'m-'); %当 ke=0.5,用紫色实线
    case 9
        %当 ke=0.75,用蓝色虚线
        plot(t,y,'b--');
    otherwise
        disp('输入错误');
end
end
legend('ke=0.3','ke=0.6','ke=0.9'); %添加图例
title('二阶系统阶跃输入响应'); %添加标题
xlabel('时间/秒'); %添加 x 坐标
ylabel('输出响应'); %添加 y 坐标
grid on

```

运行程序，输出效果如图 2-1 所示。

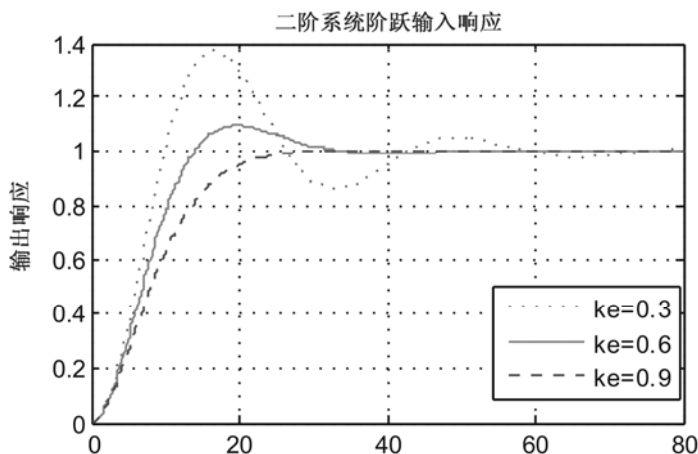


图 2-1 二阶系统单位阶跃响应曲线

2.1.3 循环结构

循环结构可以用来重复计算某一任务，在 MATLAB 中提供了 for 和 while 两种实现循环函数的结构，它们分别实现固定和不定的次数计算。

for 循环可以执行固定或者预定的次数循环计算，其调用格式如下：

for i=表达式

表达式语句, ..., 表达式语句

end

i 为循环指针变量，表达式通常是一个向量，如 $i=m:s:n$ ，m 为初始值，n 为终止值，s 为增量（也即步长）。如果 s 为正值，要求 $m < n$ ，如果 s 为负值，要求 $m > n$ 。当循环指针变量 i 的值等于向量的某个元素时，执行循环体内的表达式语句，如果循环指标变量 i 的值超过向量的上

限，则循环结束。

如果计算矩阵型数据，则需要利用两层 for 循环，即

```
for i=表达式
    for j=表达式
        表达式语句,...,表达式语句
    end
end
```

在一些特殊场合，可能需要利用多层循环，用户根据上面的两层循环类推即可得到多层循环的结构。在调节循环结构的时候，循环每一步都计算的量可以放到循环结构前面进行计算，如此可以节省不必要的计算。

如果用户在循环计算的过程中要求满足条件时中止循环计算，可以在 for 循环里面加入 break 函数来中止程序，使用如下格式：

```
for i=表达式
    表达式语句
    if 表达式
        break;
    end
end
```

与 break 函数类似的一个函数是 return 函数。这个函数可以使当前正在运行的函数正常结束，同时返回调用它的函数继续运行后面的程序，或者返回调用它的环境，如命令窗和 MATLAB 程序文件中。该函数通常用于函数文件中。用户可以对输入的参数进行判断，如果参数不符合要求，就调用 return 语句中止当前程序的运行，并返回到调用它的函数或环境，这样可以省去一些不必要的计算量。

【例 2-4】从自然数 1 开始累加，加数为自然数的质数因子最小数，直至累加和达到 100 时停止累加，返回累加和于停止的位置。

这里预设程序计算到自然数 100。其实现的 MATLAB 程序代码如下：

```
>> clear all;
s=0;           %初始化累加变量
for i=1:100;
    f=factor(i); %对 i 进行质因数分解
    fm=min(f);  %获得所有因数中的最小值
    fn(i)=fm;
    s=s+fm;     %累加求和
    if s>=100;  %检查 s 是否大于等于 100
        break; %满足条件停止程序
    end
end
i,s,fn
```

运行程序，输出如下：

```
i =
    7
s =
```

```

140
fn =
    1     4     9    16    25    36    49

```

可见截止的自然数是 7，累加和是 140。fn 记录了相应的最小质数。

与 for 循环不同的是，while 循环将以上不确定的次数执行循环结构。其调用格式如下：

while 表达式

语句命令

end

当表达式值为真时，执行循环体内的指令语句；当表达式为假时，循环结束。在使用 while 循环的时候，一定要注意避免表达式为恒真的情况，否则程序陷入死循环。

【例 2-5】获得 100 以内所有加法表达式，其中加数有两个，它们是在 1~99 之间的数，同时限制第 1 个加数小于第二个加数。要求把所有加法算式在命令窗口中显示。

其实现的 MATLAB 程序代码如下：

```

>> clear all;
a=1;          %初始化加数 a
b=2;          %初始化加数 b
N=0;          %加法算式个数的计数器,这里初始化为 0
while a<100
    if a+b<100.5;
        disp([num2str(a),'+',num2str(b),'=',num2str(a+b)]) %输出加法算式
        b=b+1;
        N=N+1;
    else
        a=a+1;
        b=a+1;
    end
end
N

```

运行程序，输出如下：

```

1+2=3
1+3=4
.....
49+50=99
49+51=100
N =

    2450

```

等价地利用两重循环结构来计算这个问题，相应程序如下：

```

>> clear all;
a=1;          %初始化加数 a
N=0;          %加法算式个数的计数器,这里初始化为 0
while a<100
    b=a+1;
    while a+b<100.5;
        disp([num2str(a),'+',num2str(b),'=',num2str(a+b)]) %输出加法算式
        b=b+1;
    end
    a=a+1;
end

```



```

        N=N+1;
    end
    a=a+1;
end
N
1+2=3
1+3=4
.....
49+50=99
49+51=100
N=
    2450
    
```

其中外层循环是变化 a ，里面一层循环是变化 b 。两层循环输出的结果和一层循环输出的结果一样，这里不再显示。

和 `for` 循环一样，存在数组定义时，需要预先定义出数组的内存空间，这样可以理解避免在程序计算过程中出现动态内存浪费的问题。

2.2 M 文件

M 文件是包含 MATLAB 代码的文件。

2.2.1 M 文件类型

M 文件按其内容和功能可以分为脚本 M 文件和函数 M 文件两大类。

1. 脚本 M 文件

它是许多 MATLAB 代码按顺序组成的命令序列集合，不接受参数的输入和输出，与 MATLAB 工作空间共享变量空间。

它一般用来实现一个相对独立的功能，比如对某个数据进行某种分析、绘图，求解某个已知条件下的微分方程等。用户可以通过在命令窗口中直接输入文件名来运行脚本 M 文件。

通过脚本 M 文件，用户可以把为实现一个具体功能的一系列 MATLAB 代码书写在一个 M 文件中，每次只需要输入文件名即可运行脚本 M 文件中的所有代码。

2. 函数 M 文件

它也是实现一个单独功能的代码块，但与脚本 M 文件不同的是函数 M 文件需要接受参数输入和输出，函数 M 文件中的代码一般只处理输入参数传递的数据，并把处理结果作为函数输出参数返回给 MATLAB 工作空间中的指定接收变量。

因此，函数 M 文件具有独立的内部变量空间。在执行函数 M 文件时，要指定输入参数的实际取值，而且一般要指定接收输出结果的工作空间变量。

MATLAB 提供的许多函数就是用函数 M 文件编写的，尤其是各种工具箱中的函数，用户可以打开这些 M 文件来查看。实际上，对于特殊应用领域的用户，如果积累了充分的专业领域应用的函数，就可以组建自己的专业领域工具箱了。

通过函数 M 文件，用户可以把实现一个抽象功能的 MATLAB 代码封装成一个函数接口，在以后的应用中重复调用。

【例 2-6】对于函数“`rank`”，该函数用来求矩阵的秩，查看该函数的代码。

在 MATLAB 的命令窗口中输入:

```
type rank
```

将显示函数 “rank” 的代码如下:

```
function r = rank(A,tol)
%RANK    Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol = max(size(A)) * eps(norm(A)).
%
%   Class support for input A:
%       float: double, single

%   Copyright 1984-2004 The MathWorks, Inc.
%   $Revision: 5.11.4.3 $   $Date: 2004/08/20 19:50:33 $

s = svd(A);
if nargin==1
    tol = max(size(A')) * eps(max(s));
end
r = sum(s > tol);
```

2.2.2 M 文件的结构

例 2-6 中显示的结果是 rank 函数 M 文件的全部内容, 清楚地显示了一般的 M 文件包括的各部分结构。

从例 2-6 显示结果可以看到, MATLAB 中 M 文件一般包括以下四部分结构:

(1) 函数声明 (Function Definition Line), 这一行只出现在函数 M 文件的第一行, 通过 function 关键字表明此文件是一个函数 M 文件, 并指定函数名、输入和输出参数。例如:

```
%RANK    Matrix rank.
```

(2) H1 行, 这是帮助文字的第一行 (the first help text line), 给出 M 文件帮助最关键的信息。当用 lookfor 查找某个单词相关的函数时, lookfor 只在 H1 行中搜索是否出现指定的单词。例如:

```
%   RANK(A) provides an estimate of the number of linearly
```

(3) 帮助文字, 这部分对 M 文件有更加详细的说明, 经常解释 M 文件实现的功能, M 文件中出现的各变、参数的意义, 以及创作版权信息等。例如:

```
%   $Revision: 5.11.4.3 $   $Date: 2004/08/20 19:50:33 $
```

(4) M 文件正文, 这部分出现的位置比较灵活, 主要是用来注释 M 文件正文的具体运行过程, 以方便阅读和修改, 经常穿插在 M 文件正文中间。例如:

```
s = svd(A);
if nargin==1
    tol = max(size(A')) * eps(max(s));
end
r = sum(s > tol);
```

2.2.3 M 文件的创建

虽然一般脚本 M 文件可以包括上述四部分结构中除去“函数声明行”以外的三部分，但在实际应用中，脚本 M 文件经常仅仅由 M 文件正文和注释部分构成。正文部分实现功能，注释部分则给出每一块代码的功能说明。下面通过示例讲述脚本 M 文件的创建。

【例 2-7】建立一个 M 命令文件，将变量 a、b 的值互换。

首先打开 M 文件编辑器，输入以下代码：

```
>> clear all;
a=1:9;
b=[11 21 31;12 22 32;13 23 33];
c=a; a=b;b=c;
a
b
```

然后保存文件名为“li2_7.m”，即完成了文件的建立。

在 MATLAB 的命令窗口中输入“li2_7”并回车，将会执行该命令文件，并在命令窗口中显示：

```
a =
    11    21    31
    12    22    32
    13    23    33
b =
     1     2     3     4     5     6     7     8     9
```

函数 M 文件的命名一般习惯和函数名一致，比如例 2-6 函数声明行 `function r = rank(A,tol)`，表明函数名为 `rank`，因此此函数 M 文件一般保存为 `rank.m`，可以通过 `rank` 语句调用该文件；否则，如果函数名和文件名不一致，函数调用就需要通过文件名和与函数声明中对应的参数列表来实现。

编写好的 M 文件，相当于 MATLAB 提供的命令，可以在命令行进行函数调用。但要注意，要求被调用的函数对应的 .m 文件必须在 MATLAB 路径下。

2.3 MATLAB 函数的调用与参数传递

MATLAB 中函数的调用方法与 C 语言中的调用方法相似，可以在控制窗口以命令行形式调用，也可以在 M 函数中调用。而 MATLAB 的参数传递一般是值传递，另外，MATLAB 支持多个返回参数。

2.3.1 函数的调用

当从命令行或 M 文件调用另外一个 M 文件时，MATLAB 把函数解析成伪码（Pseudocode）并存储在内存中。这样在下次调用此函数时，就不必再次对此函数解析了。除非用 `clear` 函数清除或退出 MATLAB，否则，此伪码一直会驻留在内存。

可以使用下面几种方法清除内存中的伪码：

- (1) `clear functionname`：清除内存文件名的伪码。
- (2) `clear functions`：清除 M 函数的伪码。

(3) `clear all`: 清除内存中所有变量和函数。

1. 函数调用顺序

当多个函数有相同的函数名时, MATLAB 将根据函数的位置及类型按顺序调用。MATLAB 调用函数的顺序遵循以下原则:

(1) 变量。在进行函数名匹配之前, MATLAB 先在当前工作空间查找是否存在以此为名的变量。若存在, 则以为是变量同时结束查找。

(2) 子函数。子函数比相同路径上其他的 M 函数和重载函数的优先级要高。

(3) 私有函数。

(4) 类的构造函数。

(5) 重载函数。

(6) 当前路径上的函数。

(7) 嵌套函数。

第(3)条到第(7)条中的函数都可以是下面5种类型之一: M 函数、内嵌函数、P 伪码、MEX 文件和 Simulink model (MDL 文件)。对于这5种类型的函数, 也有如下调用顺序:

(1) MATLAB 的内嵌函数。

(2) MEX 文件。

(3) MDL 文件。

(4) P 伪码文件。

(5) 用户的 M 文件。

用 `which` 命令可以查询 MATLAB 会调用哪个函数。比如:

```
>> which pie3
D:\MATLABR2008a\toolbox\matlab\specgraph\pie3.m
```

2. 函数调用语法

命令行方式调用函数的语法如下, 函数名后跟参数列表, 函数名和参数以及各参数之间用空格符隔开:

`functionname in1 in2...inN`

命令行的函数语法比较简单易写, 但是, 缺点是不能为函数的返回参数赋值。

命令行调用函数的简单例子如下:

```
save mydata.mat x y z
clear length width depth
```

函数式的语法和其他编程语言相似, MATLAB 的特别之处在于一个函数可以返回多于一个参数。

只有一个返回值的函数语法如下:

`out=functionname(in1, in2, ..., inN)`

若函数返回多于一个参数, 参数之间用逗号或空格隔开, 然后用括号 ([]) 把所有参数括起来:

`[out1, out2, ... outN]=functionname(in1, in2, ..., inN)`

例如:

```
copyfile(srcfile,'..\mytests','writable')
[x1,x2,x3,x4]=deal(A{:})
```

用函数式调用函数时, 对参数表中的参数进行赋值, 例如: 下面的表达式为 `polyeig` 函数

的参数 A0、A1 和 A2 赋值。

```
e=polyeig(A0, A1, A2)
```

而用命令行式调用函数时，参数以文字字符的形式传递，下面的表达为 save 函数传递了字符串形式的参数'mydata.mat'、'x'、'y'和'z'。

```
save mydata.mat x y z
```

【例 2-8】以例子来说明两种传递参数方式的不同。

```
>> %先函数式调用 disp 函数
```

```
A=pi;
```

```
disp(A); %函数式调用 disp
```

```
% 再以命令行形式调用 disp A,传递字符串参数'A';
```

```
A=pi;
```

```
disp A; %命令形式调用 disp
```

显示结果如下：

```
3.1416 与字符'A'
```

以命令行执行 disp A 时，会把 A 当做字符串，而不是变量，所以会显示'A'，而非 A 所代表的内容。

【例 2-9】用命令行形式和函数式分别为 strcmp 函数传递两个有相同内容的字符串，查看结果。

```
>> str1='MATLAB'; str2='MATLAB';
```

```
strcmp(str1,str2) %函数式调用
```

```
ans =
```

```
1 (相等)
```

用函数式调用 strcmp 函数时，得到的结果为 1，表示两字符串相等（相同）。

```
>> str1='MATLAB'; str2='MATLAB';
```

```
strcmp str1 str2 %命令形式调用
```

```
ans =
```

```
0 (不相等)
```

用命令行方式调用 strcmp 函数时，得到的结果为 0，即不相等（相同），因为在这种情况下，由于命令行是调用函数，参数是以字符串形式传递的，所以，strcmp 函数比较的是字符串'str1'和'str2'，而非 str1 和 str2 的值'MATLAB'。

2.3.2 参数传递

对函数进行调用时，返回参数的个数可以少于函数定义时的返回参数个数，但是不可以多于。比如，一个函数定义有 N 个返回参数，但是调用时，可以使用 0 到 N 个返回参数。不需要的返回参数被丢弃。函数调用时，按照函数定义行指定的顺序来返回参数。

【例 2-10】参数返回的顺序示例。

```
function [a b c]=myfun(x,y)
```

```
b=x*y;
```

```
a=100;
```

```
c=x.^2;
```

上例先返回 100，然后是 $x*y$ ，最后是 $x.^2$ ，虽然先计算得到了 $x*y$ ，但是参数列表中 a 在 b 前面，故先返回 a 的值。

当返回参数个数少于函数定义的返回参数个数时，尤其要注意上面的顺序。如调用时有一

个返回参数:

```
a=myfun(x, y)
```

此时仅仅返回 100 而不是 $x*y$ 的值, b 和 c 的值被丢弃。如调用时无任何返回参数:

```
>> myfun(5,6)
ans=
    100
```

1. 检查参数个数

与 C 语言一样, MATLAB 可获得输入参数个数信息, 并根据不同输入参数个数完成不同的功能。MATLAB 中用 `nargin` 和 `nargout` 函数获取函数调用时输入参数和输出参数的个数。

【例 2-11】对输入参数判断, 然后完成不同的任务。

```
function c=testarg(a,b)
if(nargin==1)
    c=a.^2;
elseif(nargin==2)
    c=a+b;
end
```

2. 任意个输入/输出参数

利用 `varargin` 和 `varargout` 函数可以传递任意数目的输入参数和输出参数。MATLAB 将所有的输入参数打包成一个细胞数组, 而输出参数需要自己编写代码打包成细胞数组, 以便 MATLAB 将输出参数传递给调用者。

使用 `varargout` 来返回可选的参数值有下面的两种定义方式:

```
function varargout=myfun(vin1, vin2,...)
function [vout1, vout2,..., varargout]= myfun(vin1, vin2,...)
```

用第一种定义形式定义的函数, 其函数体内创建了 `varargout` 细胞数组。此细胞数组的元素及其顺序决定了函数被调用时 MATLAB 如何为可选的返回值赋值。这种情况下 `varargout` 是函数定义行中等号左边唯一的变量, MATLAB 将 `varargout{1}` 赋值给第一个返回参数, 将 `varargout{2}` 赋值给第二个返回参数, 依次类推。

用第二种定义形式定义的函数, 除了 `varargout`, 函数定义行中还有其他的返回参数。此时 MATLAB 先为调用函数时最左边返回参数赋值, 然后按照顺序为 `varargout` 数组赋值。

【例 2-12】演示使用 `varargout`: 输入参数的矩阵必须是 2 列, 而行数可以任意, 第 1 列为 x 坐标集合, 第 2 列为 y 坐标集合。

```
function [varargout]=testvar2(arrayin)
for k=1:nargout
    varargout{k}=arrayin(k,:); %元胞数组赋值
end
```

函数把每一行的坐标对分离成单独的 $[xi \ yi]$, 这些坐标值组成一个坐标向量。

调用 `testvar2`:

```
>> a=[11 12;21 22;31 32;41 42;51 52];
>> [p1,p2,p3,p4,p5]=testvar2(a)
p1 =
    11    12
p2 =
    21    22
```

```
p3 =
    31    32
p4 =
    41    42
p5 =
    51    52
```

【例 2-13】说明使用 varargin：下面的函数可以接受任意多个二维向量，然后用直线将以这些二维向量为坐标的点连接起来。

```
function testvar(varargin)
for k=1:length(varargin)
    x(k)=varargin{k}(1); %元胞矩阵索引
    y(k)=varargin{k}(2);
end
xmin=min(0,min(x));
ymin=min(0,min(y));
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)
```

testvar 函数可以接受任意多的参数，比如，下面调用方式都是可以的：

```
testvar([2 3],[1 5],[4 8],[6 5],[4 2],[2 3])
testvar([-1 0],[3 -5],[4 2],[1 1])
```

由于 varargin 用一个元胞数组包含了所有的输入参数，所以，可以用元胞数组的索引来获取每个参数的值。例如：

```
y(n)=varargin{n}(2)
```

元胞数组索引有以下两部分，用花括号 {} 的索引表示元胞数组元胞的索引，用圆括号 () 的索引表示某元胞的内容索引。像上面的代码，表示式 {i} 访问 varargin 元胞数值的第 i 个值，表达式 (2) 访问第 i 个细胞的第二个内容。

varargin 和 varargout 必须出现在参数列表的最后，前面可以是任意个输入或输出参数。下面的 varargin 和 varargout 位置都是正确的：

```
function [out1, out2]=exampleM(a, b, varargin)
function [i, j, varargout]=exampleN(x1, y1, x2, y2, flag)
```

在嵌套函数中使用任意个参数时特别注意，因为在嵌套函数中的 varargin, varargout, nargin 和 nargout 的具体含义可能会发生混乱。varargin 和 varargout 是变量，和其他变量一样，遵循 MATLAB 的变量作用域原则。另外，注意嵌套函数与所有外部函数共享工作空间。若 nargin 或 nargout 出现在嵌套函数中，则代表传递给这个函数的输入参数或输出参数的个数，不管这个函数是否嵌套函数。若嵌套函数需要得到外部函数的 nargin 值和 nargout 值，可以将此作为参数传递。nargin 和 nargout 为函数，会记录被调用时的输入参数和输出参数。

若函数体中对输入参数进行修改，则需将此参数也列入输出参数，这样调用函数就能得到修改后的值。例如：

```
function [text, offset]=readText(filestart, offset)
```

调用 readText 函数时，读取某文件一行，则必须记录此次文件的偏移量 offset，以便下次调用时能获取开始读取的位置。但是每次 readText 函数调用结束后，offset 变量的赋值就从内存中清除了。为保存 offset 的值，采用上述调用方法，将 offset 作为返回值。

另外，MATLAB 提供了一个 inputParser 类来处理传递给 M 文件函数中不同类型的参数。

2.4 MATLAB 的编程技巧

MATLAB 是一种科学计算语言，但同时也具有和 C、FORTRAN 等高级语言相类似的语言特性，能方便地实现程序控制。利用 MATLAB 的程序控制功能，可以将有关 MATLAB 命令编成程序存储在一个文件中（M 文件），然后运行该文件，MATLAB 就会自动依次执行文件中的命令，直到全部命令执行完毕。

编程时，首先要考虑到变量初值或者变量类型改变时，程序的应对能力，即程序的鲁棒性，因此出色的程序要具有较好的例外处理机制。此外，还要考虑程序的执行效率。

一些编程技巧能提高程序的执行效率，因此，掌握一些 MATLAB 的编程技巧也非常必要。下面对常用的 MATLAB 编程技巧进行说明。

2.4.1 线性索引技巧

在 MATLAB 中，数组元素是按列排列的，也就是相当于把数组的所有元素按列向量依次排成一个很长的列向量，这就是线性索引。通过线性索引可以对数组中的元素提取、更改，也可以进行运算。

利用索引可以在编程时减少代码。例如，求数组 A 的所有元素的和，可以用求和函数 sum。对矩阵 A，由于函数 sum 是按列求和，可以用 sum(sum(A))来求出所有元素的和，但这并不是最好的方法，特别是数组的维数很大的时候。其实按照线性索引只要用 sum(A(:))，就可以求出所有元素的和，不管数组 A 是多少维的。

2.4.2 嵌套计算技巧

一个程序的执行速度取决于它所调用的子程序个数以及采用的算法。通常希望子程序越少越好，算法效率越高越好。

嵌套计算是一种具有较小时间复杂度的算法。

【例 2-14】嵌套计算与直接求值的比较示例。考虑下面两个多项式，其中第一个式为嵌套表达式。

$$p1(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$
$$p2(x) = ((a_3x + a_2)x + a_1)x + a_0$$

下面用具体程序进行说明，创建函数 li2_14。

```
function li2_14()
N=100000;
a=[1:N];
x=1;
tic           %时钟计算开始
p1=sum(a.*x.^[N-1:-1:0]);
p1
toc           %时钟计算结束,并获得执行 p1 时间
tic           %嵌套时钟计算开始
p2=a(1);
for i=2:N
    p2=p2*x+a(i);
```



```
end
p2
toc      %嵌套计算结束,并获得执行 p2 时间
tic      %自带函数求值时钟计算
p3=polyval(a,x)
toc      %自带函数求值时钟计算结束,并获得执行 p3 时间
```

在命令窗口中输入 li2_14 并回车, 输出结果如下:

```
p1 =
    5.0001e+009
Elapsed time is 0.010837 seconds.
p2 =
    5.0001e+009
Elapsed time is 0.002403 seconds.
p3 =
    5.0001e+009
Elapsed time is 0.129729 seconds.
```

可见, 嵌套算法耗时最少, 而用 polyval 求值执行速度最慢。

采用嵌套算法不仅能够提高执行效率, 而且此方程具有更强的解决问题的能力。

【例 2-15】嵌套计算与非嵌套计算的比较示例。求 poisson 分布的有限项和, 形如

$$S(M) = \sum_{n=0}^M \frac{\lambda^n}{n!} e^{-\lambda}$$

由概率论知识可知, 当 M 很大时, 上式的值趋近于 1。其实现的 MATLAB 程序代码如下:

```
function li2_15()
r=80;
M=160;
p=exp(-r);
s1=0;
for k=1:M          %嵌套式
    p=p*r/k;
    s1=s1+p;
end
s1
s2=0;
for k=1:M          %非嵌套式
    p=r^k/factorial(k);
    s2=s2+p;
end
s2=s2*exp(-r);
s2
```

在命令窗口中输入 li2_15 并回车, 输出结果如下:

```
s1 =
    1.0000
s2 =
    5.5406e+034
```

由结果可知, 嵌套方法的结果非常接近真实值, 而非嵌套的方法根本无法得到正确的结果。

2.4.3 循环计算技巧

循环计算可按照给定的条件,重复执行指定的语句。MATLAB 用于实现循环计算的语句有 for 语句和 while 语句。

对于循环的使用需要注意以下几点:

(1) 尽量避免使用循环。在 MATLAB 编程中,采用循环会降低程序的执行速度,应尽量避免使用,可以用其他方式,如向量运算等代替。

(2) 为了得到最大的速度,在 for 循环被执行之前,应预先分配数组。否则,在 for 循环内每执行一次命令,对数组重新分配一次内存,这样会降低 MATLAB 的执行效率。

(3) 优先考虑内联 (inline) 函数,矩阵运算应该尽量采用 MATLAB 的内联函数,因为内联函数是由更底层的编程语言 C 语言构造的,其执行速度显然快于使用循环的矩阵运算。

(4) 应用 MEX 技术。尽管采用了很多措施,但执行速度仍然很慢,比如说耗时的循环是不可避免的,这样就应该考虑用其他语言,如 C 或 FORTRAN 语言。按照 MEX 技术要求的格式编写相应部分程序,然后通过编译连接,形成在 MATLAB 中可以直接调用的动态链接库 (DLL) 文件,这样可以显著地加快运算速度。

2.4.4 利用“:”和 end 技巧

在编程时,运用“:”和 end 有时可以大量地减少代码,提高运算效率。

【例 2-16】编写一个函数 $B=li2_16(A, m)$, 把输入的矩阵 A 循环上移 m 行。输入参数 A 是矩阵, m 是正整数。输出参数是矩阵。

对于下面的矩阵 A 和 B 。把矩阵 A 循环上移一行得到矩阵 B 。

$$A = \begin{bmatrix} 1 & 5 & 9 & 13 & 17 \\ 2 & 6 & 10 & 14 & 18 \\ 3 & 7 & 11 & 15 & 19 \\ 4 & 8 & 12 & 16 & 20 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 6 & 10 & 14 & 18 \\ 3 & 7 & 11 & 15 & 19 \\ 4 & 8 & 12 & 16 & 20 \\ 1 & 5 & 9 & 13 & 17 \end{bmatrix}$$

其 MATLAB 代码如下:

```
function B=li2_16(A,m)
% 函数 b=li2_16(A, m)把矩阵 A 循环移动 m 行
% 输入参数 A 是矩阵
% 输入参数 m 是标量
if (floor(m)~=m)|(length(m)~=1)|(m<1)      %如果参数 m 不是正整数
    error('上移的行数应为正整数');          %显示出错信息
end
B=A;      %初始化输出变量
for k=1:m %移动 m 次
    B(1:end-1,:)=A(2:end,:); %每次上移一行
    B(end,:)=A(1,:);
    A=B;      %准备下一次上移
end
```

在函数中,由于用了“:”和 end,不必关心输入的矩阵是多少行、多少列。这个函数可以改写成一个把矩阵上、下、左、右都可循环移动的函数。

在命令窗口中输入如下命令，输出如下：

```
>> A=reshape(1:20,4,5),B=li2_16(A,1)
A =
     1     5     9    13    17
     2     6    10    14    18
     3     7    11    15    19
     4     8    12    16    20
B =
     2     6    10    14    18
     3     7    11    15    19
     4     8    12    16    20
     1     5     9    13    17
```

2.4.5 使用全局变量技巧

全局变量是指在不同的工作空间以及基本的工作空间中可以共享的变量。用户只需要在主程序或者任何子程序中声明一个或多个全局变量，则函数和主程序中都可以直接引用它们，采用如下格式：

```
global v1 v2 ... vn
```

表达式中各变量之间用空格隔开。

使用全局变量时要注意以下几点：

(1) 它可以在主程序和函数之间不需要经过输入或输出变量直接传递数据。但要注意在函数调用中使用它们时，调用结束后全局变量在工作空间中仍然存在。

(2) 两个或多个函数也可以共有一个全局变量，只要同时在这些函数中用 `global` 语句加以定义即可。

(3) 使用全局变量时必须十分小心，最好把全局变量名取得长一些或全部用大写，以免与函数中的局部变量重名。如果重名，容易出错致命错误，所以，使用全局变量不是一个好的编程方法。

(4) 一旦变量被声明为全局的，则在任何声明它的地方都可以对它进行修改。这在一定程度上破坏了子程序的独立性。如果全局变量被多个子程序修改，则用户很难知道全局变量的确切值，这使得程序的可读性大大下降。

【例 2-17】 本例用以说明全局变量的声明及函数的传递。

建立子程序 `li2_17A.m` 和主程序 `li2_17.m`，同时在子程序 `li2_17A.m` 以及主程序 `li2_17.m` 中定义全局变量 `d`，其 MATLAB 的程序代码如下：

```
% 子程序代码
function x=li2_17A(t,d)
global d           %声明全局变量
t(find(t==0))=eps; %防止分母出现 0 项
x=sin(pi*t/d)./(pi*t*d);
% 主程序代码
function li2_17()
global d
d=2;
b1=-2;
```

```

b2=2;
t=b1+[0:100]/100*(b2-b1);
% 通过全局变量传递参数
plot(t,li2_17A(t));

```

在命令窗口中输入 li2_17 函数，输出效果如图 2-2 所示。

如果在子程序 li2_17A 中修改全局变量的值，则变量声明时即对其进行赋值。例如 li2_17A 中的声明语句改为：global d=1，则运行后得图 2-3 所示的结果。

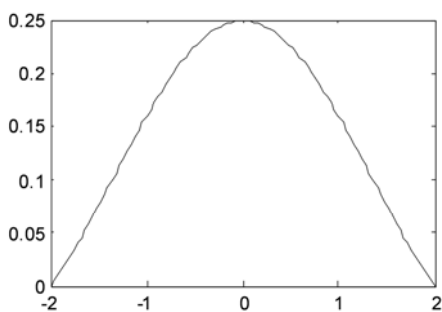


图 2-2 输出效果

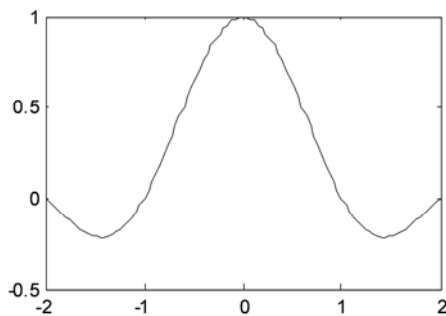


图 2-3 全局变量修改后的显示结果

2.4.6 使用例外处理机制技巧

优秀的程序员能够指导用户如何使用他编写的程序，而且在用户使用不当时，能够给出错误的提示信息，并引导用户正确使用函数。

一般而言，对于输入参量小于设定个数的情形，MATLAB 内置程序一般会对未赋值参数作默认处理。典型的例子是 plot(x, y) 函数，plot(y) 默认的 x 坐标是 [0, 1, 2, ...] 序列。

程序员在编写程序的时候也应当注意处理这种情况。采用 nargin 函数可以判断输入参量的个数，从而设定未被指定的输入参数的值或者直接报错。

【例 2-18】利用 nargin 函数，实现两个多项式的相加，并具有一定的报错功能。

其实现的 MATLAB 程序代码如下：

```

function p=li2_18(a,b)    %求多项式 a, b 和 p
if nargin==1
    b=zeros(4,1);
elseif nargin==0
    error('empty input');    %输入参数个数为 0,报错
end
a=a(:)';
b=b(:)';
na=length(a);
nb=length(b);
p=[zeros(1,nb-na) a]+[zeros(1,na-nb) b]; %多项式相加

```

在命令窗口中输入正确的参数，调用该函数，输出正确的结果，如下：

```

>> a=[11 22 33 44];
>> li2_18(a)
ans =
    11    22    33    44

```

在 MATLAB 命令窗口中输入错误的参数，调用该函数，输出报错结果，如下：

```
>> li2_18()
??? Error using ==> li2_18 at 5
empty input
```

可见，用户输入参数不符合要求，会作默认处理或报错，使得程序具有更强的适应性。

2.4.7 倒序法技巧

倒序法就是在循环的时候，把变量从最大的元素向最小的元素倒序计算。这种方法也可以使得系统不需要重新分配存储区，从而减少程序运行的时间。

【例 2-19】创建脚本文件如下。

```
clear all;
x=0:0.01:200;
tic;
for i=1:length(x)
    y1(i)=x(i).^3;
end;
toc;
clear all;
tic;
x=0:0.01:200;
for i=length(x):-1:1
    y2(i)=x(i).^3;
end;
toc
```

把文件保存为 li2_19.m。在 MATLAB 命令窗口中输入：

```
>> li2_19
```

输出如下：

```
Elapsed time is 1.065715 seconds.
Elapsed time is 0.008333 seconds.
```

从运行的结果可以看出，用倒序所用时间大为减少。

2.4.8 向量法处理技巧

MATLAB 是一种解释性语言，对每一条命令，都要进行代码分析、语法检查、生成可执行命令后，才可以执行。这通常要消耗时间，特别是遇到循环时，这种时间的消耗会明显增加。因此在遇到一些重复运算时，可以考虑能否把要进行的运算化为向量或矩阵的形式，以提高运行的效率。

【例 2-20】创建脚本文件如下：

```
clear all;
x=0:0.01:200;
tic;
y=x.^3;
toc;
clear all;
x=0:0.01:200;
```

```
tic;
for i=1:length(x)
    y(i)=x(i).^3;
end;
toc
```

把文件保存为 li2_20.m。在 MATLAB 命令窗口中输入：

```
>> li2_20
```

输出如下：

```
Elapsed time is 0.007285 seconds.
```

```
Elapsed time is 1.016038 seconds.
```

从运行的结果可以看出，使用向量化操作和循环操作所用时间相差非常大（文件运行时间根据所用的机器不同，会出现不同结果）。

2.5 插值和拟合

插值与拟合是来源于实际、又广泛应用于实际的两种重要方法。随着计算机的不断发展及计算机水平的不断提高，它们已在国民生产和科学研究等方面扮演着越来越重要的角色。

表 2-1 总结了 MATLAB 中所具有的曲线拟合和插值函数。

表 2-1 曲线拟合和插值函数

曲线拟合和插值函数	功 能
<code>polyfit(x, y, n)</code>	对描述 n 阶多项式 $y=f(x)$ 的数据进行最小二乘曲线拟合
<code>interp1(x, y, x0)</code>	一维线性插值
<code>interp1(x, y, x0, 'spline')</code>	一维三次样条插值
<code>interp1(x, y, x0, 'cubic')</code>	一维三次插值
<code>interp2(x, y, Z, xi, yi)</code>	二维线性插值
<code>interp2(x, y, Z, xi, yi, 'cubic')</code>	二维三次插值
<code>interp2(x, y, Z, xi, yi, 'nearest')</code>	二维最近邻插值

2.5.1 一维插值

插值定义对数据点之间函数的估值方法，这些数据点是由某些集合给定。当用户不能很快地求出所需中间点的函数值时，插值是一个有价值的工作。例如，当数据点是某些实验测量的结果或是过长的计算过程时，就有这种情况。

最简单的插值例子或许就是 MATLAB 的作图。按默认状态，MATLAB 用直线连接所用的数据点以作图。这个线性插值猜测中间值落在数据点之间的直线上。当然，当数据点个数的增加和它们之间的距离减小时，线性插值就更精确。例如，代码如下：

```
>> clear all;
x1=linspace(0,2*pi,60);
x2=linspace(0,2*pi,6);
plot(x1,sin(x1),'r',x2,sin(x2));
xlabel('x'); ylabel('sin(x)');
```

```
title('线性插值');
```

运行程序，效果如图 2-4 所示。

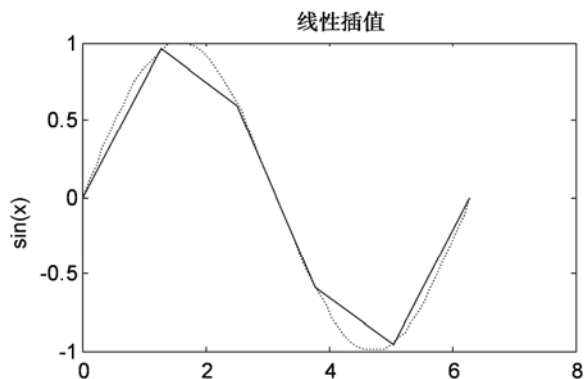


图 2-4 线性插值

图 2-4 是 \sin 函数的两个图，一个在数据点之间用 60 个点，它比另一个只用 6 个点更光滑和更精确。

插值要作决策。根据所作的假设，有多种插值。而且，可以在一维以上空间中进行插值，即如果有反映两个变量函数的插值， $z=f(x, y)$ ，那么就可以在 x 之间和在 y 之间找出 z 的中间值进行插值。MATLAB 在一维函数 `interp1` 和在二维函数 `interp2` 中，提供了多种插值选择，其中各个函数将在下面进行介绍。

为了说明一维插值，24h 内，1h 测量一次室外温度。数据存储在两个 MATLAB 变量中。其实现的 MATLAB 程序代码如下：

```
>> clear all;
hours=1:24;           %定义时间参数
%定义温度
temps=[5 9 11 10.5 15 18 22 21 30 31 30.5 29 28 20 19 18 17 20 25 27 24 18 15 14];
plot(hours,temps,'rp',hours,temps); %观察温度
title('室外温度');
xlabel('hours'); ylabel('摄氏度');
```

运行程序，效果如图 2-5 所示。

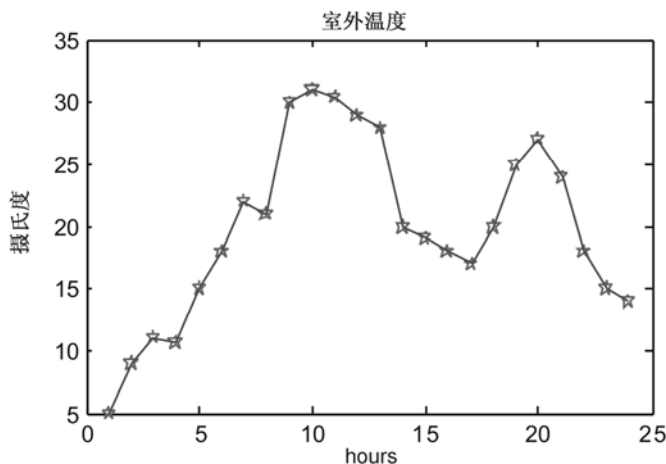


图 2-5 在线性插值下的室外温度曲线

由图 2-5 可知, MATLAB 绘制出了数据点线性插值的直线。为了计算在任意给定时间的温度, 用户可试着对可视的图作解释。另外一种方法, 就是使用 `interp1` 函数。

其实现的 MATLAB 程序代码如下:

```
>> t=interp1(hours,temps,10.8)    %计算在 10.8 小时的温度
t =
    30.6000
>> t=interp1(hours,temps,6.4)      %计算在 6.4 小时的温度
t =
    19.6000
>> t=interp1(hours,temps,[1.5 3.7 8.2 11.4])
t =
    7.0000    10.6500    22.8000    29.9000
```

`interp1` 的默认用法是由 `interp1(x, y, x0)` 来描述, 这里 x 是独立变量 (横坐标), y 是应变量 (纵坐标), x_0 是进行插值的一个数值数组。另外, 该默认的使用假定为线性插值。

若不采用直线连接数据点, 用户可采用某些更光滑的曲线来拟合数据点。最常用的方法是用一个三阶多项式, 即三次多项式, 来对相继数据点之间的各段进行建模, 每个三次多项式的前两个导数与该数据点相一致。这种类型的插值被称为三次样条或简称为样条。函数 `interp1` 也能执行三次样条插值。

```
>> t=interp1(hours,temps,10.8,'spline')    %计算在 10.8 小时的温度
t =
    30.6722
>> t=interp1(hours,temps,6.4,'spline')      %计算在 6.4 小时的温度
t =
    20.0016
>> t=interp1(hours,temps,[1.5 3.7 8.2 11.4],'spline')
t =
    7.0057    10.2859    22.2729    29.8248
```

注意: 样条插值得到的结果, 与上面所示的线性插值结果不同。因为插值是一个估计或猜测的过程, 其意义在于, 应用不同的估计规则导致不同的结果。

一个最常用的样条插值是对数据平滑。也就是, 给定一组数据, 使用样条插值在更细的间隔求值。例如, 其代码如下:

```
>> h=1:0.1:24;                %计算每 1/10 小时的温度
t=interp1(hours,temps,h,'spline');
plot(hours,temps,'rp',hours,temps,h,t); %比较结果图
title('室外温度曲线');
xlabel('hour'); ylabel('摄氏度');
```

运行程序, 效果如图 2-6 所示。

在图 2-6 中, 蓝线是线性插值, 绿线是平滑的样条插值, 标有红色“星号”的是原始数据。如要求在时间轴上有更细的分辨率, 并使用样条插值, 有一个更平滑、但不一定更精确地对温度的估计。尤其要注意的是, 在数据点, 样条解的斜率不突然改变。作为这个平滑插值的回报, 三次样条插值要求更大量的计算, 因为必须找到三次多项式以描述给定数据之间的特征。

在讨论二维插值之前, 了解 `interp1` 所强制的两个强约束是很重要的。首先, 用户不能要求有独立变量范围以外的结果, 例如, `interp1(hours, temps, 24.5)` 导致一个错误, 因为 `hours` 是在 1~24 之间的变化。其次, 独立变量必须是单调的, 即独立变量在值上必须总是增加或总是减小。

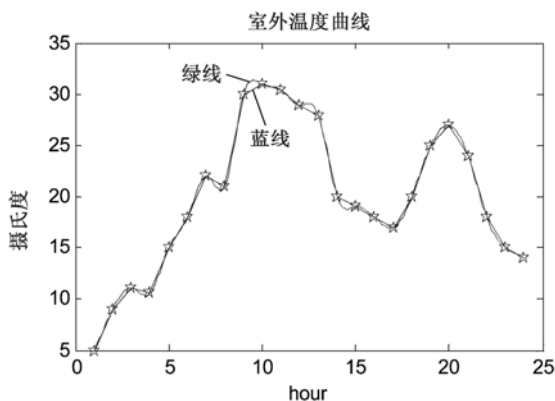


图 2-6 在不同插值下的室外温度曲线

2.5.2 二维插值

二维插值是基于与一维插值同样的思想。然而，正如名字所隐含的，二维插值是对两变量的函数 $z=f(x, y)$ 进行插值。它的调用格式如下：

$ZI = \text{interp2}(X, Y, Z, XI, YI)$

$ZI = \text{interp2}(Z, XI, YI)$

$ZI = \text{interp2}(Z, \text{ntimes})$

$ZI = \text{interp2}(X, Y, Z, XI, YI, \text{method})$

式中： X, Y, Z 为样本数据矩阵； XI, YI, ZI 为插值数据矩阵； ntimes 为在元素间插入扩充数据的回归次数； method 为选取插值的方法。插值的方法有以下 4 种：

- (1) 邻近插值， $\text{method}=\text{nearest}$;
- (2) 双线条插值， $\text{method}=\text{linear}$;
- (3) 样条插值， $\text{method}=\text{spline}$;
- (4) 立方插值， $\text{method}=\text{cubic}$ 。

【例 2-21】 interp2 函数应用示例。

```
>>clear all;
[X,Y] = meshgrid(-3:.25:3);
Z = peaks(X,Y);
[XI,YI] = meshgrid(-3:.125:3);
ZI = interp2(X,Y,Z,XI,YI,'nearest');
mesh(X,Y,Z), hold, mesh(XI,YI,ZI+15)
hold off
axis([-3 3 -3 3 -5 20])
```

运行程序，效果如图 2-7 所示。

【例 2-22】 设人们对平板上的温度分布估计感兴趣，给定的温度值取自平板表面均匀分布的格栅。

采集了下列的数据：

```
width=1:5;      %定义平板宽度(例如 x 方向)
depth=1:3;      %定义平板深度(例如 y 方向)
temps=[82 81 80 82 83;78 65 60 79 81;83 82 81 85 87]
temps =
```

82	81	80	82	83
78	65	60	79	81
83	82	81	85	87

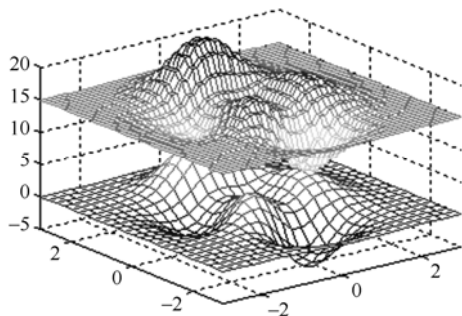


图 2-7 interp2 函数显示效果

如同标引点上测量一样，矩阵 **temps** 表示整个平板的温度分布。**temps** 的列与下标 **depth** 或 **y** 维相联系，行与下标 **width** 或 **x** 维相联系。为了估计在中间点的温度，必须对它们进行辨识。其实现的 MATLAB 程序代码如下：

```
>> clear all;
wi=1:0.2:5;
d=2;
zlin=interp2(width,depth,temps,wi,d); %双线性插值
zcu=interp2(width,depth,temps,wi,d,'cubic'); %立方插值
plot(wi,zlin,'-',wi,zcu);
xlabel('板宽');ylabel('摄氏度');
title(['板深度 d=' num2str(d),'处的温度']);
```

运行程序，效果如图 2-8 所示。

另一种方法，可以在两个方向插值。先在三维坐标绘制出原始数据，看一下该数据的粗糙程度。其实现的 MATLAB 程序代码如下：

```
>> mesh(width,depth,temps);
xlabel('板宽');ylabel('板深度');zlabel('摄氏度');
```

运行程序，效果如图 2-9 所示。

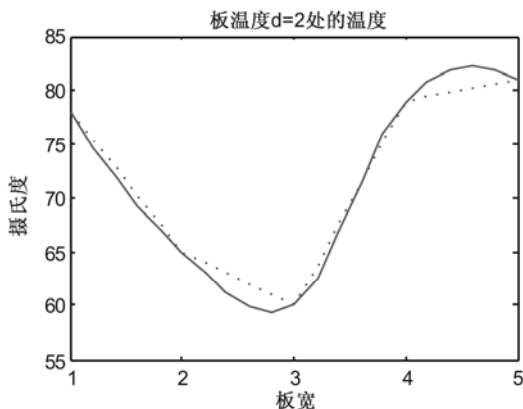
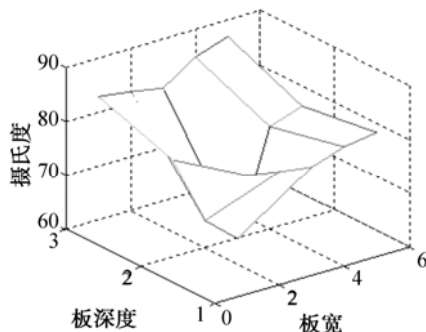
图 2-8 在深度 $d=2$ 处的平板温度

图 2-9 平板温度

然后以平滑数据在两个方向上插值。其实现的 MATLAB 程序代码如下：

```
>> di=1:0.2:3;
wi=1:0.5:5;
[wi,di]=meshgrid(1:0.5:5,1:0.2:3);
zcu=interp2(width,depth,temps,wi,di,'cubic');
mesh(wi,di,zcu);
xlabel('板宽'); ylabel('板深度'); zlabel('摄氏度');
axis('ij');
```

运行程序，效果如图 2-10 所示。

可选的参数 `method` 可以是 `linear`，`cubic` 或 `nearest`。在这种情况下，`cubic` 不意味着三次样条，而是使用三次多项式的另一种算法。`linear` 方法是线性插值，仅用作连接图上数据点。`nearest` 方法只选择最接近各估计点的粗略数据点。在所有的情况下，假定独立变量 `x` 和 `y` 是线性间隔和单调的。

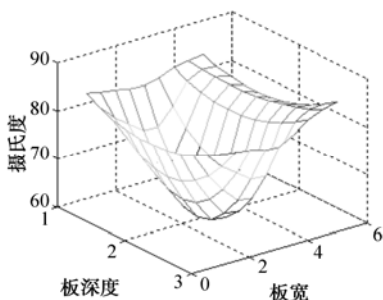


图 2-10 二维插值后的平板温度

2.5.3 高维插值

在最简单的用法中，`spline` 获取数据 `x` 和 `y` 以及期望值 `xi`，寻找拟合 `x` 和 `y` 的三次样条内插多项式，然后，计算这些多项式，对每个 `xi` 的值，寻找相应的 `yi`。例如，代码如下：

```
>> x=0:12;
y=tan(pi*x/25);
xi=linspace(0,12);
yi=spline(x,y,xi);
plot(x,y,'p',xi,yi);
title('样条插值');
```

运行程序，效果如图 2-11 所示。

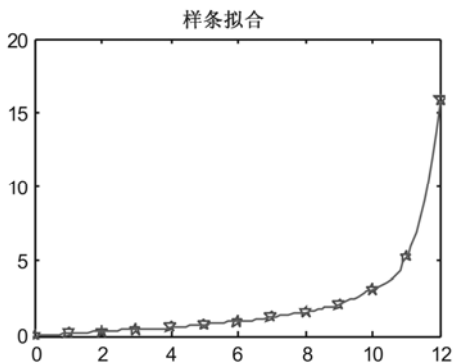


图 2-11 样条拟合效果

这种方法适合于只需要一组内插值的情况。不过,如果需要从相同数据集中获取另一组内插值,再次计算三次样条系数是没有意义的。在这种情况下,可以调用仅带前两个参数的 `spline`:

```
>> pp=spline(x,y)
pp =
    form: 'pp'
    breaks: [0 1 2 3 4 5 6 7 8 9 10 11 12]
    coefs: [12x4 double]
    pieces: 12
    order: 4
    dim: 1
```

当采用这种方式调用时, `spline` 返回一个称之为三次样条的 `pp` 形式或分段多项式形式的数组。这个数组包含了对于任意一组所期望的内插值和计算三次样条所必需的全部信息。给定 `pp` 形式,函数 `ppval` 计算该三次样条。如:

```
>> yi=ppval(pp,xi);
计算先前计算过的同样的 yi。
类似地有
```

```
>> xi2=linspace(10,12);
>> yi2=ppval(pp,xi2);
```

运用 `pp` 形式,在限定的更细区间[10 12]内,再次计算该三次样条。

```
>> xi3=10:16
xi3 =
    10    11    12    13    14    15    16
>> yi3=ppval(pp,xi3)
yi3 =
    3.0777    5.2422   15.8945   44.0038   98.5389  188.4689  322.7629
```

这表明,可在计算三次多项式所覆盖的区间外,计算三次样条。当数据出现在最后一个断点之后或第一个断点之前,则分别运用最后一个或第一个三次多项式来寻找内插值。

上述给定的三次样条 `pp` 形式,存储了断点和多项式系数,以及关于三次样条表示的其他数据结构。当要计算三次样条表示时,必须把 `pp` 形式分解成它的各个表示段。在 MATLAB 中,通过函数 `unmkpp` 完成这一过程。运用上述 `pp` 形式,该函数给出如下结果:

```
>> [breaks, coefs, npolys, ncoefs]=unmkpp(pp)
breaks =
    0    1    2    3    4    5    6    7    8    9   10   11   12
coefs =
    0.0007   -0.0001    0.1257         0
    0.0007    0.0020    0.1276    0.1263
    0.0010    0.0042    0.1339    0.2568
    0.0012    0.0072    0.1454    0.3959
    0.0024    0.0109    0.1635    0.5498
    0.0019    0.0181    0.1925    0.7265
    0.0116    0.0237    0.2344    0.9391
   -0.0083    0.0586    0.3167    1.2088
    0.1068    0.0336    0.4089    1.5757
   -0.1982    0.3542    0.7967    2.1251
    1.4948   -0.2406    0.9102    3.0777
```

```
1.4948    4.2439    4.9136    5.2422
npolys =    12
ncoefs =     4
```

这里 `breaks` 是断点, `coefs` 是矩阵, 它的第 i 行是第 i 个三次多项式, `npolys` 是多项式的数目, `ncoefs` 是每个多项式系数的数目。注意, 这种形式非常一般, 样条多项式不必是三次, 这对于样条的积分和微分是很有益的。

给定上述分散形式, 函数 `mkpp` 恢复了 `pp` 形式。

```
>> pp=mkpp(breaks,coefs)
pp =
    form: 'pp'
  breaks: [0 1 2 3 4 5 6 7 8 9 10 11 12]
   coefs: [12x4 double]
 pieces: 12
  order: 4
   dim: 1
```

因为矩阵 `coefs` 的大小确定了 `npolys` 和 `ncoefs`, 所以 `mkpp` 不需要 `npolys` 和 `ncoefs` 去重构 `pp` 形式。`pp` 形式的数据结构仅在 `mkpp` 中给定为 `pp=[10 1 npolys break(:)'ncoefs coefs(:)']`。前两个元素出现在所有的 `pp` 形式中, 它们作为确认 `pp` 形式向量的一种方法。

2.5.4 最小二乘拟合

逼近实验数据的基本方法就是拟合方法, 其中最常用的就是最小二乘法拟合。最小二乘法适用的数学模型可以是线性函数和非线性函数。对于一组给定的数据 (x_k, y_k) , 设计和确定一个函数模型 $y=f(x)$, 使得函数 $f(x)$ 在某种误差函数标准下与左右数据点 (x_k, y_k) 之间的距离最近, 也就是说寻求一个最佳拟合函数。

最小二乘拟合是解决曲线拟合问题的常用方法。这里以线性函数为例介绍最小二乘法的原理, 该方法的基本思路是建立一个线性模型, 即

$$y=f(x)=ax+b$$

其中: a 和 b 是待定系数。通过实验数据 (x_k, y_k) 确定出系数 k 和 b 。最小误差原则是保证 y_k 和 $f(x_k)$ 之间距离的平方和最小, 因此这种方法被称为线性最小二乘法。

下面考虑系数 a 和 b 的求法。考虑下面的误差函数:

$$\Delta(a,b)=\sum_{k=1}^n[y_k-f(x_k)]^2=\sum_{k=1}^n[y_k-(ax_k+b)]^2$$

可以发现 $\Delta(a,b)$ 是关于 a 和 b 的二次函数。根据高等数学的知识, $\Delta(a,b)$ 具有最小值的必要条件是 $\frac{\partial \Delta(a,b)}{\partial a}=0$ 和 $\frac{\partial \Delta(a,b)}{\partial b}=0$, 从而有

$$\begin{cases} \sum_{k=1}^n x_k [ax_k + b - y_k] = 0 \\ \sum_{k=1}^n [ax_k + b - y_k] = 0 \end{cases}$$

求解上面关于 a 和 b 的方程组有: $a = \frac{\overline{x_k y_k} - \bar{x}_k \bar{y}_k}{\overline{x_k^2} - \bar{x}_k^2}$, $b = \bar{y}_k - a \bar{x}_k$ 。其中 \bar{x}_k 和 \bar{y}_k 分别为 x_k

和 y_k 的均值, $\overline{x_k y_k}$ 和 $\overline{x_k^2}$ 分别为 $x_k y_k$ 和 x_k^2 的均值。

相应的误差评估通常使用下面两种形式。

(1) 最小平方根误差 (均方误差):

$$\Delta_1 = \left\{ \sum_{k=1}^n [y_k - f(x_k)]^2 \right\}^{\frac{1}{2}}$$

(2) 最大偏差:

$$\Delta_2 = \max_{1 \leq k \leq n} |y_k - f(x_k)|$$

类似地, 对于多项式函数和更复杂的函数表示, 都可以利用上面的思路来确定模型中的未知系数。对于单个数学模型, 使用最小二乘法只能获得一个拟合结果, 可以计算相应的误差值, 如果误差值没有达到要求, 就要更改其他模型, 直到满足要求为止。

实际中, 一些模型可以转化为线性最小二乘问题, 如:

(1) $y = ax^m + b$, 其中 $m \neq 1$, 通过代换 $u = x^m$, 可以得到一个线性模型。

(2) $y = a \cdot e^{bx^m}$, 通过代换 $u = x^m$ 及两边取对数操作, 可以简化为线性模型。

在实际应用中, 需要考虑不同的模型, 通过误差函数值的比较从中选择拟合后误差最小的。而函数模型的设计需要根据经验来选择, 在数据规律不明显的时候, 可以通过平移、取对数或者分段处理等操作来探索数据的规律。

【例 2-23】计算线性最小二乘拟合, 处理数据见表 2-2。

表 2-2 线性最小二乘拟合数据

xk	0	1	2	3	4	5	6	7	8	9
yk	2	3.4	5.6	8	11	12.3	13.8	16	18.8	19.9

其实现的 MATLAB 程序代码如下:

```
>> clear all;
xk=0:9;
yk=[2 3.4 5.6 8 11 12.3 13.8 16 18.8 19.9];
% xk 和 yk 为离散数据
plot(xk,yk,'rp');           %画离散数据点
h=lsline                    %最小二乘拟合
xd=get(h,'XData');          % 获取直线的数据
yd=get(h,'YData');          % 获取直线的数据
a=[yd(1)-yd(2)]./[xd(1)-xd(2)] %斜率
b=yd(1)-a*xk(1)             %截距
```

a 和 b 的值如下:

```
a =
    2.0582
b =
    1.8182
```

运行程序, 效果如图 2-12 所示。

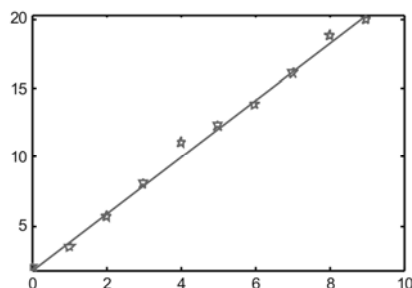


图 2-12 线性最小二乘拟合

此外，函数 `polyfit` 可以用来进行多项式拟合，如果选择 1 次多项式，那么正好对应着上面的模型。为了比较上面的结果，这里调用 `polyfit` 函数进行拟合：

```
>> p=polyfit(xk,yk,1)
```

运行程序，输出如下：

```
p =  
    2.0582    1.8182
```

这里 `p(1)` 是 1 次项系数，`p(2)` 是多项式的常数项。可以发现 `polyfit` 函数与 `lsline` 函数返回了相同的结果，利用 `polyfit` 函数可以方便地实验数据进行线性最小二乘拟合处理。关于 `polyfit` 函数将在 2.5.5 小节展开介绍。

接下来考虑一种多元线性拟合问题。这里以二元问题为例，一般的函数模型可以写为 $z=f(x,y)=ax+by+c$ ，其中 a 、 b 和 c 为待定系数。数据点 (x_k, y_k, z_k) 带入上述函数模型可以得到方程组 $ax_k+by_k+c=z_k$ ($1 \leq k \leq n$)。

上面的方程组可以写为矩阵形式 $M \cdot X=d$ ，其中：

$$M = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix}, X = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, d = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

方程 $M \cdot X=d$ 可以转化为求下面的最小二乘优化问题：

$$\min_x \frac{1}{2} \|MX - d\|_2^2$$

MATLAB 中提供了函数 `lsqlin` 计算最小二乘优化问题，其调用格式如下：

```
x = lsqlin(C,d,A,b)  
x = lsqlin(C,d,A,b,Aeq,beq)  
x = lsqlin(C,d,A,b,Aeq,beq,lb,ub)  
x = lsqlin(C,d,A,b,Aeq,beq,lb,ub,x0)  
x = lsqlin(C,d,A,b,Aeq,beq,lb,ub,x0,options)  
x = lsqlin(problem)  
[x,resnorm] = lsqlin(...)  
[x,resnorm,residual] = lsqlin(...)  
[x,resnorm,residual,exitflag] = lsqlin(...)  
[x,resnorm,residual,exitflag,output] = lsqlin(...)  
[x,resnorm,residual,exitflag,output,lambda] = lsqlin(...)
```

【例 2-24】lsqlin 函数应用示例。

```
>> C = [ 0.9501    0.7620    0.6153    0.4057
        0.2311    0.4564    0.7919    0.9354
        0.6068    0.0185    0.9218    0.9169
        0.4859    0.8214    0.7382    0.4102
        0.8912    0.4447    0.1762    0.8936];
d = [ 0.0578
      0.3528
      0.8131
      0.0098
      0.1388];
A = [ 0.2027    0.2721    0.7467    0.4659
      0.1987    0.1988    0.4450    0.4186
      0.6037    0.0152    0.9318    0.8462];
b = [ 0.5251
      0.2026
      0.6721];
lb = -0.1*ones(4,1);
ub = 2*ones(4,1);
[x,resnorm,residual,exitflag,output,lambda]=lsqlin(C,d,A,b,[],[],lb,ub);
x,lambda.ineqlin,lambda.lower,lambda.upper
```

运行程序，输出如下：

```
x =
    -0.1000
    -0.1000
     0.2152
     0.3502
lambda.ineqlin =
         0
     0.2392
         0
lambda.lower =
     0.0409
     0.2784
         0
         0
lambda.upper =
         0
         0
         0
         0
```

有的时候在测量数据中存在着一些明显误差点，也称它们为坏点。在进行拟合计算时，不希望考虑这些点。MATLAB 提供了函数 `robustfit` 来完成健壮回归分析，该函数的调用格式如下：

```
b = robustfit(X,y)
b = robustfit(X,y,wfun,tune)
b = robustfit(X,y,wfun,tune,const)
```


[b,stats] = robustfit(...)

【例 2-25】处理带有一个异常点的数据，比较最小二乘拟合值与稳健拟合差异。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
    randn('state',2007);           %设置随机数状态
    x=[1:8]';                       %产生 x 数据
    y=8+3*x+randn(8,1);             %产生 y 数据
    y(end)=2;                       %设置最后一个数据为异常数据
    plot(x,y,'rp','MarkerFaceColor','k'); %画出数据点
    h=lsline;                        %添加最小二乘结果
    set(h,'lineStyle','-','Color','r'); %改变线型和颜色
    P=polyfit(x,y,1);                %一次多项式拟合
    b=robustfit(x,y);                %稳健拟合
    hold on;
    plot(x,b(1)+b(2)*x,'k');         %绘制稳健拟合曲线
    set(gca,'FontSize',12);           %字体设置
    xlim([min(x),max(x)]);
    P,b
```

运行程序，输出如下，效果如图 2-13 所示。

```
P =
    0.5016    15.6436
b =
    8.1077
    3.0170
```

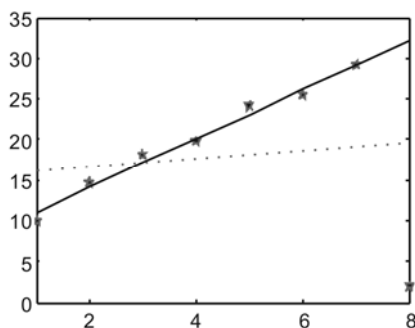


图 2-13 稳健拟合与一般拟合的比较

可见函数 robustfit 没有考虑最后那个坏点，对数据的拟合程度好些，而函数 lsline 和 polyfit 把所有数据点都考虑进去进行最小二乘计算，拟合结果受到异常值影响，直线偏向异常值一侧。在实际应用中可以根据不同需要选择拟合方式。

2.5.5 多项式拟合

一般多项式拟合的目标是找出一组多项式系数 $a_i, i=1, 2, \dots, n+1$ ，使得多项式： $\psi(x) = a_1x^n + a_2x^{n+1} + \dots + a_nx + a_{n+1}$ 能够较好地拟合原始数据。和前面介绍插值算法不同，多项式拟合并不能保证每个样本点都在拟合的曲线上，但能使得整体的拟合误差较小。多项式拟合可以通过 MATLAB 提供的 polyfit 函数实现。该函数的调用格式如下：

`p=polyfit(x, y, n)`

其中： x 和 y 为原始的样本点构成的向量； n 为选定的多项式阶次，得出的 p 为多项式系数按降幂排列得出的行向量，可以用符号运算工具箱中的 `poly2sym` 函数将其转化成多项式形式，也可以使用 `polyval` 函数求取多项式的值。下面通过例子演示多项式拟合函数的使用方法和优缺点。

【例 2-26】`polyfit` 函数应用示例。

```
>> clear all;
x = (0: 0.1: 2.5)';
y = erf(x);
p = polyfit(x,y,6)
f = polyval(p,x);
table = [x y f y-f]
```

运行程序，输出如下，效果如图 2-14 所示。

```
p =
    0.0084    -0.0983     0.4217    -0.7435     0.1471     1.1064     0.0004
table =
         0         0     0.0004    -0.0004
    0.1000    0.1125    0.1119     0.0006
    0.2000    0.2227    0.2223     0.0004
    0.3000    0.3286    0.3287    -0.0001
    0.4000    0.4284    0.4288    -0.0004
    0.5000    0.5205    0.5209    -0.0004
    0.6000    0.6039    0.6041    -0.0002
    0.7000    0.6778    0.6778     0.0000
    0.8000    0.7421    0.7418     0.0003
    0.9000    0.7969    0.7965     0.0004
    1.0000    0.8427    0.8424     0.0003
    1.1000    0.8802    0.8800     0.0002
    1.2000    0.9103    0.9104    -0.0000
    1.3000    0.9340    0.9342    -0.0002
    1.4000    0.9523    0.9526    -0.0003
    1.5000    0.9661    0.9664    -0.0003
    1.6000    0.9763    0.9765    -0.0002
    1.7000    0.9838    0.9838     0.0000
    1.8000    0.9891    0.9889     0.0002
    1.9000    0.9928    0.9925     0.0003
    2.0000    0.9953    0.9951     0.0002
    2.1000    0.9970    0.9969     0.0001
    2.2000    0.9981    0.9982    -0.0001
    2.3000    0.9989    0.9991    -0.0003
    2.4000    0.9993    0.9995    -0.0002
    2.5000    0.9996    0.9994     0.0002

>> x = (0: 0.1: 5)';
y = erf(x);
f = polyval(p,x);
plot(x,y,'p',x,f,'-')
axis([0 5 0 2])
```

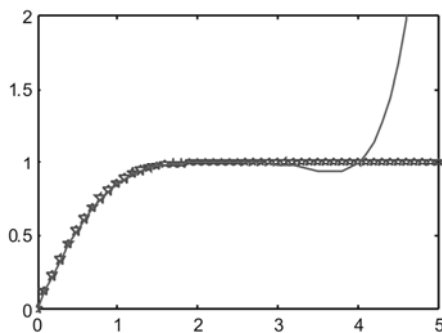
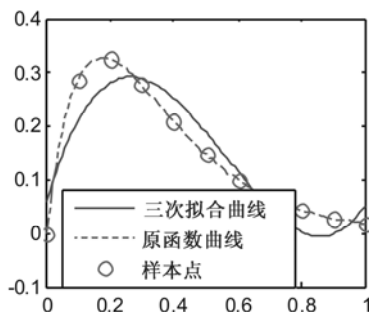


图 2-14 多项式拟合效果

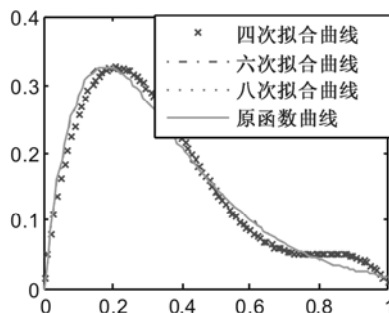
【例 2-27】 已知的数据点来自函数，根据生成的数据试用多项式拟合的方法在不同的阶次下进行拟合，并观察拟合效果，找出合适的阶次。

可以用下面的语句得出拟合该数据的三次多项式并绘制出拟合曲线，如图 2-15 (a) 所示。

```
>> clear all;
x0=0:0.1:1;
y0=(x0.^2-3*x0+5).*exp(-5*x0).*sin(x0);
p3=polyfit(x0,y0,3);      %三次多项式拟合
vpa(poly2sym(p3),10)      %显示多项式
x=0:0.01:1;
ya=(x.^2-3*x+5).*exp(-5*x).*sin(x);
y1=polyval(p3,x);
plot(x,y1,x,ya,x0,y0,'o');
legend('三次拟合曲线','原函数曲线','样本点');
```



(a) 三次多项式拟合



(b) 其他次数的多项式拟合

图 2-15 多项式拟合效果

运行程序，输出如下多项式：

```
ans =
2.839962923*x^3-4.789842696*x^2+1.943211631*x+.5975248921e-1
```

从拟合结果可以看出，结果还是相当差的。一种很显然的解决方法是提高拟合多项式的次数，下面就不同的次数进行拟合，最终得出如图 2-15 (b) 所示的拟合效果。

```
>> p4=polyfit(x0,y0,4);
y4=polyval(p4,x);
p6=polyfit(x0,y0,6);
y6=polyval(p6,x);
```

```
p8=polyfit(x0,y0,8);
y8=polyval(p8,x);
plot(x,y4,'x',x,y6,'-',x,y8,'-',x,ya,'-');
legend('四次拟合曲线','六次拟合曲线','八次拟合曲线','原函数曲线');
```

从该例的拟合效果看,当拟合多项次数等于8时,多项式拟合曲线与原函数曲线已经基本重合,拟合效果已经比较令人满意。这时,拟合多项式可以由下面的语句显示出来,可以用该多项式在[0,1]区间内近似模拟原函数模型。

```
>> vpa(poly2sym(p8),5)
ans =
-8.2586*x^8+43.566*x^7-101.98*x^6+140.22*x^5-125.29*x^4+74.450*x^3-27.672*x^2+4.9869*x+
.42037e-6
```

2.5.6 非线性拟合

MATLAB 提供了两个求非线性最小二乘拟合函数,即 `lsqcurvefit` 和 `lsqnonlin`。函数 `lsqcurvefit` 的调用格式如下:

```
x = lsqcurvefit(fun,x0,xdata,ydata)
x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub)
x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub,options)
x = lsqcurvefit(problem)
```

函数 `lsqnonlin` 函数的调用格式如下:

```
x = lsqnonlin(fun,x0)
x = lsqnonlin(fun,x0,lb,ub)
x = lsqnonlin(fun,x0,lb,ub,options)
x = lsqnonlin(problem)
```

【例 2-28】使用非线性函数模型。

应用函数 $y = f(x) = c_1 + c_2 e^{-0.02c_3 x}$ (其中 c_1 、 c_2 和 c_3 是待定系数)来拟合如表 2-3 所列的数据。

表 2-3 拟合数据

x	1	2	3	4	5	6	7	8	9	10
y	3.5	3.0	2.6	2.3	2.1	1.9	1.7	1.6	1.5	1.4

应用 `lsqnonlin` 函数时,首先要定义函数模型,根据表达式定义 `li2_28A` 函数模型,根据表达式其源代码如下:

```
function y= li2_28 A(x,xd); %定义 lsqcurvefit 拟合函数的输入函数
y=x(1)+x(2)*exp(-0.02*x(3)*xd);
其实现的 MATLAB 程序代码如下:
>> clear all;
xd=1:10; % 读入数据
yd=[3.5 3.0 2.6 2.3 2.1 1.9 1.7 1.6 1.5 1.4];
c0=[0 1 1]; %初始值
c=lsqcurvefit('li2_28 A',c0,xd,yd) %进行非线性拟合
plot(xd,yd,'rp'); %利用原始数据绘图
```

```
hold on;
xp=linspace(min(xd),max(xd),200); %生成原始数据范围内的等间隔采样点
yp= li2_28 A (c,xp);               %利用拟合系数计算因变量数值
plot(xp,yp)                         %绘制拟合曲线
```

运行程序，输出拟合如下，效果如图 2-16 所示。

```
c =
    1.0992    2.9910   11.2454
```

接下来使用函数 `lsqnonlin` 拟合上面的数据，此时自定义的 `li2_28` 的源代码如下：

```
function y=li2_28B(x); % 定义 lsqnonlin 拟合函数的输入函数
x=1:10;               % 读入数据
y=[3.5    3.0  2.6  2.3  2.1  1.9  1.7  1.6  1.5  1.4];
y=yd-[x(1)+x(2)*exp(-0.02*x(3)*xd)];
```

其实现的 MATLAB 程序代码如下：

```
>> clear all;
c0=[0 1 1]; %初始值
c=lsqnonlin('fit_model_b',c0) %进行非线性拟合
```

运行程序，输出如下：

```
c =
    1.0992    2.9910   11.2454
```

可见两个函数的输出结果是一样的，但是它们拟合函数的输入方式不同。此外初始值 `x0` 的选取也会影响结果，可以套用上面的格式来解决自己的问题。

多元非线性拟合问题可以利用多元非线性回归分析。

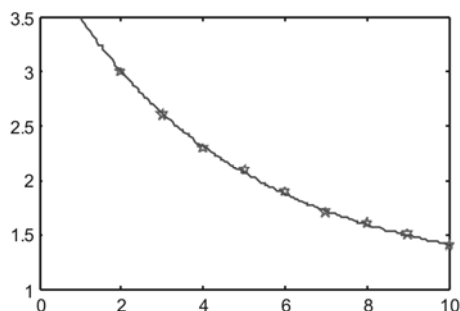


图 2-16 非线性数据拟合

第3章 MATLAB的符号计算

MATLAB 自产生之日起就在数值计算应用方面独占鳌头，广受各专业计算人员的青睐。但是由于在数学、力学、物理等各科研、工程应用中还经常会遇到符号运算的问题，因此一部分 MATLAB 用户还不得不同时掌握一些符号计算语言，如 Mathematic、Maple、Mathcad 等，这就给使用 MATLAB 语言进行工程运算带来了一些不便。为了解决这个问题，MathWorks 公司于 1993 年从加拿大滑铁卢大学购入了 Maple 的使用权，并以此为基础利用 Maple 的函数库开发了 MATLAB 语言的又一个重要的工具箱——符号计算工具箱。从此 MATLAB 便集数值计算、符号计算和图形可视化三大基本功能于一体，成为在数学计算领域中功能强大、操作简单、最受用户喜爱的语言。

3.1 符号计算的基础

3.1.1 符号计算的基本概念

MATLAB 数值运算的对象是数值，而 MATLAB 符号运算的对象则是非数值的符号对象。通过 MATLAB 的符号运算功能，可以求解科学计算中数学问题的符号解析表达精确解，这在自然科学与工程计算的理论分析中有极其重要的作用与实用价值。

1. 符号对象

何谓符号对象？符号对象（Symbolic Object）是 Symbolic Math Toolbox 定义的一种新的数据类型（sym 类型），用来存储代表非数值的字符符号（通常是大写或小写的英文字母及其字符串）。符号对象可以是符号常量（符号形式的数）、符号变量、符号函数以及各种符号表达式（符号数学表达式、符号方程与符号矩阵）等。

2. 创建符号对象与函数命令

在一个 MATLAB 程序中，作为符号对象的符号常量、符号变量、符号函数以及符号表达式，首先必须用函数命令 sym、syms 加以规定，即创建或建立。

1) sym 函数

sym 函数用于创建一个符号对象。其调用格式如下：

$S = \text{sym}(A)$ 或 $S = \text{sym}('A')$ ：由 A 来建立一个符号对象 S，其类型为 sym 型。如果 A（不带单引号）是一个数字（值）或数值矩阵或数值表达式，则输出是将数值对象转换成的符号对象。如果 A（带单引号）是一个字符串，输出则是将字符串转换成的符号对象。

$S = \text{sym}(A, \text{flag})$ 或 $S = \text{sym}('A', \text{flag})$ ：同 $S = \text{sym}(A)$ 。只不过转换成的符号对象应符合 flag 格式。flag 可取以下选项：

‘d’ —— 最接近的十进制浮点精确表示；

‘e’ —— 带（数值计算时）估计误差的有理数表示；

‘f’ —— 十六进制浮点表示；

‘r’ —— 为默认设置，是最接近有理表示的形式。这种形式是指用两个正整数 p、q 构成的

p/q 、 $p*\pi/q$ 、 \sqrt{p} 、 2^p 、 10^q 表示的形式之一。

`S=sym('A',flag)`: 功能同 `S=sym('A')`。只不过转换成的符号对象应按 `flag` 指定的要求。`flag` 可取以下“限定性”选项:

‘positive’——限定 A 为正的实型符号变量;

‘real’——限定 A 为实型符号变量;

‘unreal’——限定 A 为非实型符号变量。

2) 函数 syms

`syms` 函数用于创建多个符号对象。其调用格式如下:

`syms s1 s2 s3 flag`: 建立 3 (多) 个符号对象: `s1`、`s2`、`s3`。指定的要求即按 `flag` 取的“限定性”选项同上。

3. 符号常量

符号常量是一种符号对象。数值常量如果作为函数 `sym` 的输入参量, 这就建立了一个符号对象——符号常量, 即看上去是一个数值量, 但它已是一个符号对象了。创建的这个符号对象可以用 `class` 函数来检测其数据类型。

`class` 的调用格式如下:

`str=class(object)`: 这个函数命令用来测试建立的操作对象为何种操作对象类型、是否为符号对象类型 (即 `sym` 类型)。执行后返回指代数据对象类型的字符串。

4. 符号变量

变量是程序设计语言的基本元素之一。`MATLAB` 数值运算中, 变量是内容可变的数据。而 `MATLAB` 符号运算中, 符号变量是内容可变的符号对象。符号变量通常是指一个或几个特定的字符, 不是指符号表达式, 虽然可以将一符号表达式赋值给一个符号变量。符号变量有时也叫做自由变量。符号变量与 `MATLAB` 数值运算的数值变量名称的命名规则相同。

在 `MATLAB` 中, 可以用函数 `sym` 或 `syms` 来建立符号变量。

5. 符号表达式、符号函数与符号方程

已经介绍过 `MATLAB` 的数学表达式有两类: 数字表达式与符号表达式。在 `MATLAB` 符号运算中, 符号表达式是由符号常量、符号变量、符号函数用运算符或专用函数连接而成的符号对象。符号表达式有两类: 符号函数与符号方程。符号函数不带等号, 而符号方程是带等号的。

在 `MATLAB` 中, 同样用 `sym` 函数来建立符号表达式。

6. 函数 findsym

在微积分、函数表达式化简、解方程中, 确定自变量是必不可少的。在不指定自变量的情况下, 按照数学常规, 自变量通常都是小写英文字母, 并且为字母表末尾的几个如 `t`、`w`、`x`、`y`、`z` 等。在 `MATLAB` 中, 可以用函数 `findsym()` 按这种数学习惯来确定一个符号表达式中的自变量, 这对按照特定要求进行某种计算是非常有实用价值的。

函数 `findsym` 的调用格式如下:

`findsym(f,n)`: 这种格式的功能是按数学习惯确定符号函数 `f` 中的 `n` 个自变量。当指定的 `n=1` 时, 从符号函数 `f` 中找出在字母表中与 `x` 最近的字母; 如果有两个字母与 `x` 的距离相等, 则取较后的一个。当输入参数 `n` 默认时, 函数命令将给出 `f` 中所有的符号变量。

`findsym(e,n)`: 这种格式的功能是按数学习惯确定符号方程 `e` 中的 `n` 个自变量, 其余功能同上。

7. 符号矩阵

在前面已经介绍了数组与矩阵的概念，现在来说明 MATLAB 中符号矩阵的概念。元素是符号对象（非数值符号的字符符号即符号变量与符号形式的数即符号常量）的矩阵叫做符号矩阵。符号矩阵既可以构成符号矩阵函数（没有等号），也可以构成符号矩阵方程（有等号），它们都是符号表达式。

数值矩阵与符号矩阵的 MATLAB 表达式的书写特点都是：元素必须用一对方括号括起来，行之间用分号分隔，一行的元素之间用逗号或空格分隔。

3.1.2 符号表达式的创建

1. 创建符号常量

符号常量是不含有变量的符号表达式，符号常量常常与整数很难区分，例如：

【例 3-1】利用 sym 函数来创建符号常量。

```
>> a=sym('cos(2*x)')    %创建符号常量
a =
cos(2*x)
>> f=simplify((3*4-2)/5+1) %化简表达式
f =
3
>> isstr(f)    %如果 f 为字符型,返回 1,否则返回 0
ans =
0
```

其中：f 代表数字型符号常量 3，但不是字符串型。

2. 创建符号变量和符号表达式

在 MATLAB 中，符号变量和符号表达式可以通过函数 sym 和 syms 来创建。

【例 3-2】应用 sym 函数创建符号变量及符号表达式。

```
>> sym1=sym('MATLAB')
sym1 =
MATLAB
>> sym2=sym('1/9')    %创建符号变量
sym2 =
1/9
>> f=sym('a*x^4+b*x^3+c*x^2+d*x+e') %创建符号表达式
f =
a*x^4+b*x^3+c*x^2+d*x+e
```

【例 3-3】应用 syms 函数创建符号变量及符号表达式。

```
>> syms x
>> f=sin(x)+cos(x)
f =
cos(x) + sin(x)
>> syms x y
>> f=sin(x+y)+cos(x-y)
f =
cos(x - y) + sin(x + y)
```


3.2 符号矩阵的生成

在 MATLAB 中创建符号矩阵的方法和创建数值矩阵的形式很相似，只不过要用到符号定义函数 `sym`。下面介绍使用此函数创建符号函数的几种形式。

3.2.1 使用 `sym` 函数创建符号矩阵

使用 `sym` 函数直接创建符号矩阵和直接创建数值矩阵的方法几乎完全相同。矩阵元素可以是任何不带符号的符号表达式；各符号表达式的长度可以不同；矩阵元素之间可用空格或逗号分隔。

【例 3-4】使用 `sym` 函数创建符号矩阵示例。

```
>> a=sym('[1/s+x sin(x) cos(x)^2/(b+x);8 exp(x^3+y^3) log(tan(y));11 7 cos(y*2)]')
```

运行程序，输出符号矩阵如下：

```
a =  
[ 1/s+x,      sin(x), cos(x)^2/(b+x)]  
[      8, exp(x^3+y^3),  log(tan(y))]  
[    11,      7,      cos(y*2)]
```

3.2.2 将数值矩阵转化为符号矩阵

在 MATLAB 中，数值矩阵不能直接参与符号运算，必须先转化为符号矩阵。注意不论数值矩阵的元素原先是用分数还是用浮点数表示的，转化后的符号矩阵都是以最接近的精确有理数给出。

【例 3-5】将数值矩阵转化为符号矩阵示例。

```
>> a=[3/5,sqrt(3),0.115;1.7,1/0.333,log(5)]
```

```
a =  
    0.6000    1.7321    0.1150  
    1.7000    3.0030    1.6094
```

```
>> b=sym(a)
```

```
b =  
[ 3/5, sqrt(3),      23/200]  
[17/10, 1000/333, 7248263982714163*2^(-52)]
```

3.2.3 用创建子阵的方法创建符号矩阵

用创建子阵的方法创建符号矩阵是仿照 MATLAB 的字符串矩阵的直接输入法而设计的。在这种方法中，不需要调用 `sym` 命令，但要保证同一列的各元素字符串具有相同的长度。为此，在较短字符串的前后可用空格符补充。

【例 3-6】用创建子阵的方法创建符号矩阵示例。

```
>> a=sym('[1/s+x sin(x) cos(x)^2/(b+x);8 exp(x^3+y^3) log(tan(y));11 7 cos(y*2)]')
```

```
a =  
[ 1/s+x,      sin(x), cos(x)^2/(b+x)]  
[      8, exp(x^3+y^3),  log(tan(y))]  
[    11,      7,      cos(y*2)]
```

```
>> ms=['[1/s,sin(x)'],'[1 ,exp(x)']' %其中 1 的后面用三个空格填补
```

```
ms =
[1/s,sin(x)]
[1 ,exp(x)]
>> b=[a;'[exp(-j),4,x^2+y^7]']
b =
[ 1/s+x,      sin(x), cos(x)^2/(b+x)]
[      8, exp(x^3+y^3),  log(tan(y))]]
[      11,      7,      cos(y^2)]
[ exp(-j),      4,      x^2+y^7]
```

3.3 符号的基本运算

3.3.1 符号的代数运算

与数值运算一样，符号运算也可以用+、-、*、/、^ 运算符实现符号运算。下面是一些符号表达式的四则运算示例。

【例 3-7】sym 函数应用示例。

```
a=sym('a');           %定义符号变量 a, b, c
b=sym('b');
c=sym('c');
x=5; y=-8;             %定义数值变量 x, y, z
z=11;
w=a*a+b*b+c*c          %符号运算
w =
a^2+b^2+c^2
>> w=x*x+y*y+z*z       %数值运算
w =
210
>> whos
Name      Size      Bytes    Class    Attributes
a          1x1         126     sym
b          1x1         126     sym
c          1x1         126     sym
w          1x1           8    double
x          1x1           8    double
y          1x1           8    double
z          1x1           8    double
```

从代码执行情况可以看出，定义了 3 个符号变量 a、b、c，3 个数值变量 x、y、z，w 开始为符号变量，重新被赋值变量。

应用 sym 函数还可以定义符号常量，使用符号常量进行代数运算时和数值常量进行的运算不同。下面的代码用于比较符号常量与数值常量在代数运算中的差异。

```
>> pi1=sym('pi');k1=sym(8);k2=sym('4');    %定义符号变量
pi2=pi;r1=11;r2=2;                          %定义数值变量
sin(pi1/5)                                    %符号计算
ans =
```

```
sin(1/5*pi)
>> sin(pi/5)           %数值计算
ans =
    0.5878
>> sqrt(k1+sqrt(k2))    %符号计算
ans =
    10^(1/2)
>> sqrt(r1+sqrt(r2))    %数值计算
ans =
    3.5234
```

从代码执行情况可以看出，用符号常量进行计算更像在进行数学演算，所得到的结果是精确的数学表达式，而数值计算将结果近似为一个有限小数。

【例 3-8】矩阵的代数运算。

```
>> a=sym([1/x,1/(x+1);1/(x-2),1/(x^2+1)]);
>> b=sym('[x,1;x+3,7]');
>> c=a+b           %符号矩阵的加运算
c =
[      x + 1/x,      1/(x + 1) + 1]
[ x + 1/(x - 2) + 3, 1/(x^2 + 1) + 7]
>> d=a-b           %符号矩阵的减运算
d =
[      1/x - x,      1/(x + 1) - 1]
[ 1/(x - 2) - x - 3, 1/(x^2 + 1) - 7]
>> e=a*b           %符号矩阵的乘运算
e =
[      (x + 3)/(x + 1) + 1,      7/(x + 1) + 1/x]
[ x/(x - 2) + (x + 3)/(x^2 + 1), 1/(x - 2) + 7/(x^2 + 1)]
>> f=a/b           %符号矩阵的除运算
f =
[      (4*x - x^2 + 7)/(3*x*(2*x - 1)*(x + 1)),      -(x - x^2 + 1)/(3*x*(2*x - 1)*(x + 1))]
[ (6*x^2 - x + 13)/(3*(2*x - 1)*(x^2 + 1)*(x - 2)), -(2*x + 1)/(3*(2*x - 1)*(x^2 + 1)*(x - 2))]
```

【例 3-9】有符号表达式 $e_1=acx^2y+apx^2+bcxy+bpx+cky+kp$ 与 $e_2=cy+p$ ，试计算 $e_1/e_2=?$ 与 $e_1/e_2=?$ 其实现的 MATLAB 程序代码如下：

```
>>syms a b c p k x y e1 e2;
e1=a*c*x^2*y+a*p*x^2+b*c*x*y+b*p*x+c*k*y+k*p;
e2=c*y+p;
ans1=simple(e1/e2)
ans2=simple(symdiv(e1,e2))
ans3=simple(e1*inv(e2))
ans4=simple(e1\e2)
```

回车后得到结果：

```
ans1 =
a*x^2+b*x+k
ans2=
a*x^2+b*x+k
ans3=
```

```
a*x^2+b*x+k
ans4=
1/( a*x^2+b*x+k)
```

即 $e_1/e_2=ax^2+bx+k$ 与 $e_1 \setminus e_2 = \frac{1}{ax^2+bx+k}$ 。

程序中使用的函数 `symdiv()` 与使用 “/” 运算符实现右除运算的结果相同，并且其运算规则是： $a/b=a*\text{inv}(b)$ 。

需要特别注意，对于符号对象的关系运算，只有是否“等于”的关系概念，没有“大于”、“小于”等的概念。运算符“==”、“~=”分别对算符两端的对象进行“相等”、“不等”的比较。当比较的结果为“真”时，语句执行后返回“1”；当比较的结果为“假”时，语句执行后返回“0”。

还需要特别指出，MATLAB 的符号对象无逻辑运算功能。

3.3.2 提取符号表达式中的分子与分母

如果符号表达式是一个有理分式或可以展开为有理分式，可利用 `numden` 函数来提取符号表达式中的分子或分母。其一般调用格式如下：

```
[n,d]=numden(s)
```

该函数提取符号表达式 s 的分子和分母，分别将它们存放在 n 与 d 中。

【例 3-10】符号表达式的提取分子和分母运算示例。

```
>> a=sym(0.454)
a =
    227/500
>> [n,d]=numden(a)
n =227
d =500
>> f=sym('a*x^2/(b+x)')
f =
    a*x^2/(b+x)
>> [n,d]=numden(f)
n =
    a*x^2
d =b+x
```

`numden` 函数在提取各部分之前，将符号表达式有理化后，返回所得的分子和分母。例如：

```
>> g=sym('(x^2+3)/(2*x-1)+3*x/(x-1)')
g =
    (x^2+3)/(2*x-1)+3*x/(x-1)
>> [n,d]=numden(g)
n =
    x^3+5*x^2-3
d =
    (2*x-1)*(x-1)
```

如果符号表达式是一个符号数组，`numden` 返回两个新数组 n 和 d ，其中 n 是分子数组， d 是分母数组。例如：

```
>> h=sym('[4/3,(3*x+2)/5;a/x+a/y,5*x+7]')
```

```
h =
[      4/3, (3*x+2)/5]
[  a/x+a/y,   5*x+7]
>> [n,d]=numden(h)
n =
[      4, 3*x+2]
[ a*(y+x), 5*x+7]
d =
[   3, 5]
[ x*y, 1]
```

3.4 矩阵的分解与化简

对于矩阵的分解与化简操作也是 MATLAB 符号矩阵操作的重要组成部分之一。下面对常用的分解与化简进行介绍。

3.4.1 矩阵的特征值分解

符号矩阵的特征值分解由函数 `eig` 实现。其调用格式如下：

```
[V,D] = eig(A)
[V,D] = eig(A,'nobalance')
[V,D] = eig(A,B)
[V,D] = eig(A,B,flag)
```

【例 3-11】 矩阵的特征值分解示例。

```
>> h=sym('[4/3,(3*x+2)/5;a/x+a/y,5*x+7]')
h =
[      4/3, (3*x+2)/5]
[ a/x+a/y,   5*x+7]
>> [v,lambda]=eig([h])
v =
[ -(17*x*y + 15*x^2*y + 3*x*y*((72*a*x + 72*a*y + 1445*x*y + 108*a*x^2 + 2550*x^2*y + 1125*x^3*y + 108*a*x*y)/(45*x*y))^(1/2))/(6*a*x + 6*a*y), -(17*x*y + 15*x^2*y - 3*x*y*((72*a*x + 72*a*y + 1445*x*y + 108*a*x^2 + 2550*x^2*y + 1125*x^3*y + 108*a*x*y)/(45*x*y))^(1/2))/(6*a*x + 6*a*y)]
[1, 1]
lambda =
[ (5*x)/2 - ((72*a*x + 72*a*y + 1445*x*y + 108*a*x^2 + 2550*x^2*y + 1125*x^3*y + 108*a*x*y)/(45*x*y))^(1/2)/2+25/6,0]
[ 0,(5*x)/2 + ((72*a*x + 72*a*y + 1445*x*y + 108*a*x^2 + 2550*x^2*y + 1125*x^3*y + 108*a*x*y)/(45*x*y))^(1/2)/2 + 25/6 ]
```

3.4.2 矩阵的奇异值分解

符号矩阵的奇异值分解由函数 `svd` 实现。其调用格式如下：

```
[U,S,V] = svd(X)
[U,S,V] = svd(X,0)
```

```
[U,S,V] = svd(X,'econ')
```

【例 3-12】矩阵的奇异值分解示例。

```
>> a=sym([1 2;3 4;5 6;7 8])
a =
[ 1, 2]
[ 3, 4]
[ 5, 6]
[ 7, 8]
>> [U,S,V] = svd(a)
U =
[ 0.1524832333102009078749205320739,-0.82264747222565926770893134583973,-0.39450102228
382958569858561225531, -0.37995913387759662153641102261861]
[ 0.34991837180796399544747572010146,-0.42137528768458108063095627867563,0.2427965457
0435634649493621078966, 0.80065587951006292460392413239656]
[ 0.54735351030572708302003090812903,-0.020103103143502893552981211511529,0.697909975
44277606410588441518661, -0.46143435738733598459861519693729]
[ 0.74478864880349017059258609615659,0.38116908139757529352499385565257,-0.5462054988
6330282490223501372096, 0.040737611754869681531102087159343]
S =
[ 14.269095499261482682584760829662, 0]
[ 0, 0.62682823241754057153250062673738]
V =
[ 0.64142302799507239386188743537638, 0.76718739507217698818850203859595]
[ 0.76718739507217698818850203859595, -0.64142302799507239386188743537638]
```

3.4.3 矩阵的零列空间

1. 矩阵的零空间

实现符号矩阵的零空间是 `null` 函数。其调用格式如下：

`Z = null(A)`：得到由奇异值分解所得的零空间的正交基。

【例 3-13】矩阵的零列空间示例。

```
>> a=sym([1 2;3 4;5 6;7 8])
a =
[ 1, 2]
[ 3, 4]
[ 5, 6]
[ 7, 8]
>> null(a)
ans =
[ empty sym ]
>> colspace(a)
ans =
[ 1, 0]
[ 0, 1]
[ -1, 2]
[ -2, 3]
```

2. 矩阵的列空间

实现符号矩阵的列空间求解为 `colspace` 函数。其调用格式如下：

`B = colspace(A)`

【例 3-14】矩阵的列空间示例。

```
>> A = sym([2,0;3,4;0,5])
B = colspace(A)
A =
[ 2, 0]
[ 3, 4]
[ 0, 5]
B =
[      1,      0]
[      0,      1]
[ -15/8,  5/4]
```

3.4.4 因式分解

`factor` 为符号因式分解函数，其调用格式如下：

`f = factor(n)`：输入变量 `n` 为一符号矩阵，此函数将对此矩阵的各个元素进行因式分解。如果 `S` 包含的所有元素为整数，则最佳因式分解式将被计算。为了分解大于 2^{52} 的整数，请使用 `factor(sym('N'))`。

【例 3-15】因子分解示例。

```
>> syms x
>> factor(x^8-1)
ans =
(x - 1)*(x + 1)*(x^2 + 1)*(x^4 + 1)
>> factor(sym('12345678909876543210'))
ans =
2*3^2*5*7*19*928163*1111211111
```

3.4.5 同类项合并

`collect` 为合并系数函数，该函数的调用格式如下：

`collect(S, v)`：将符号矩阵 `S` 中的各元素的 `v` 的同幂项系数合并。

`collect(S)`：对由 `findsym` 函数返回的默认变量进行同类项合并。

【例 3-16】同类项合并示例。

```
>> syms x y;
>> collect(x^3*y+x^3/y-x*y-3*x+2*y)
ans =
(y + 1/y)*x^3 + (- y - 3)*x + 2*y
```

3.4.6 分式通分

函数 `numden` 可以得到符号表达式的分子和分母，其调用格式如下：

`[N,D]=numden(A)`：转换 `A` 的各元素为分子和分母都是整系数的最佳多项式。

【例 3-17】分式通分示例。

```
>> syms x y;  
>> [n,d]=numden(x/x*y+x^2/y^2)  
n =  
x^2 + y^3  
d =  
y^2
```

3.5 符号微积分

微分和积分是微积分学研究和应用的核心，并广泛地用在许多工程学科中。MATLAB 符号工具能帮助解决许多这类问题。在 MATLAB 中，提供了函数 `diff` 来进行符号的微分和积分求导运算，而通过 `Jacobian` 函数来实现对多元符号函数的求导。而积分变换在工程、应用数学、线性系统等领域都有着重要的作用。

3.5.1 符号极限

求符号表达式的极限是比较常见的操作，MATLAB 符号数学工具箱提供求符号表达式的极限函数 `limit`。其调用格式如下：

`limit(F,x,a)`：对 x 求趋于 a 的极限，但是当左右极限不相同时，极限不存在。

`limit(F,a)`：用 `findsym(F)` 作为独立变量。

`limit(F)`：对 x 求右趋于 $a=0$ 的极限。

`limit(F,x,a,'right')`：对 x 求右趋于 a 的极限。

`limit(F,x,a,'left')`：对 x 求左趋于 a 的极限。

【例 3-18】符号极限示例。

```
>> syms x a t h;  
>> limit(sin(x)/x)  
ans =  
1  
>> limit(1/x,x,0,'right')  
ans =  
Inf  
>> limit(1/x,x,0,'left')  
ans =  
-Inf  
>> limit((sin(x+h)-sin(x))/h,h,0)  
ans =  
cos(x)  
>> v = [(1 + a/x)^x, exp(-x)];  
limit(v,x,inf)  
ans =  
[ exp(a), 0]
```

3.5.2 符号级数

求符号表达式级数时，MATLAB 提供函数 `symsum` 和 `taylor` 来实现这一操作。

1. symsum 函数

符号表达式的级数求和常常采用 symsum 函数。其调用格式如下：

`r = symsum(s, v, a, b)`: 求符号表达式 s 在变量 v 取遍 $[a, b]$ 中所有整数的和, v 省略则自动对自变量求和, a, b 同时省略时默认的求和的自变量区间为 $[0, v-1]$ 。

【例 3-19】求级数 $1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{k^2}$ 的和。

```
>> syms x k
>> r=symsum(1/k^2,1,20)    %求级数前 20 项的和
r =
17299975731542641/10838475198270720
>> r=symsum(1/k^2,5,15)    %求级数前第 5 项至第 15 项的和
r =
20365731181/129859329600
>> r=symsum(1/k^2,10,20)    %求级数前第 10 项至第 20 项的和
r =
3056206830982561/54192375991353600
```

2. taylor 函数

其调用格式如下：

`taylor(f, n, v)`: 求表达式 f 进行泰勒级数展开 n 项, v 为自变量, 默认泰勒展开式为 5 项。

【例 3-20】求表达式 $\cos(x)$ 和 e^x 泰勒级展开式。

```
>> syms x
r=taylor(cos(x),10)    %cos 泰勒展开前 10 项
r =
x^8/40320 - x^6/720 + x^4/24 - x^2/2 + 1
>> r=taylor(exp(x),10)    %(ex)泰勒展开前 10 项
r =
x^9/362880 + x^8/40320 + x^7/5040 + x^6/720 + x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
```

3.5.3 符号微分

1. diff 函数

在 MATLAB 中提供了 diff 函数实现符号微分计算。其调用格式如下：

`diff(f)`: 传回 f 对预设独立变数的一次微分值。

`diff(f, 't')`: 传回 f 对独立变数 t 的一次微分值。

`diff(f, n)`: 传回 f 对预设独立变数的 n 次微分值。

`diff(f, 't', n)`: 传回 f 对独立变数 t 的 n 次微分值。

【例 3-21】符号表达式微分。

```
>> f=sym('a*x^4+x^3+x^2-b*x-2*c')
f =
a*x^4+x^3+x^2-b*x-2*c
>> diff(a,'a')
ans =
1
>> diff(f,3)
ans =
```

```
24*a*x + 6
>> diff(f,'a',3)
ans =
0
```

函数 `diff` 也可对数组进行运算。如果 `F` 是符号向量或数组，`diff(F)` 对数组内的各个元素进行微分。

【例 3-22】`diff` 函数对符号数组进行微分。

```
>> F=sym(['a*2*x,+b^2*x;c*x^4,d*x;5,5*x'])
F =
[ a*2*x, +b^2*x]
[ c*x^4,   d*x]
[    5,   5*x]
>> diff(F)
ans =
[    2*a, b^2]
[ 4*c*x^3,  d]
[    0,  5]
```

如果 `diff` 的表达式或可变参量是数值，MATLAB 就非常巧妙地计算其数值差分。如果参量是符号字符串或变量，MATLAB 就对其表达式进行微分。

2. jacobian 函数

`jacobian` 函数用于对多元符号函数的求导。其调用格式如下：

`jacobian(f, v)`：函数用于计算数量或者向量 `f` 对于向量 `v` 的 jacob (雅可比) 矩阵，计算得到的结果的第 `i` 行第 `j` 列的数为 $\frac{df(i)}{dv(j)}$ 。当 `f` 为数量时，该函数返回的是 `f` 的梯度；`v` 为数量时，

`jacobian(f, v)` 等价于前面提到的 `diff(f, v)`。

【例 3-23】利用 `jacobian` 函数对多元符号函数求导。

```
>> a=[1,x^3+x*y+y^3;5,x*cos(x)*sin(y)]
a =
[ 1, x^3 + x*y + y^3]
[ 5, x*cos(x)*sin(y)]
>> jacobian(a,[x,y])
ans =
[          0,          0]
[ 3*x^2 + y,  3*y^2 + x]
[          0,          0]
[ cos(x)*sin(y) - x*sin(x)*sin(y), x*cos(x)*cos(y)]
```

3.5.4 符号积分

积分函数 `int(f)`，其中 `f` 是一符号表达式，它力图求出另一符号表达式 `F`，使 `diff(F)=f`。正如研究微分学所了解的，积分有不定积分、定积分、旁积分和重积分等。一般来说，积分比微分要复杂得多。积分或逆求导不一定是以封闭形式存在的，或许存在但软件也许找不到，或者软件可明显地求解，但超过内存的限制。当 MATLAB 不能找到逆导数时，它将返回未经计算的命令。

相关的 `diff` 函数符号积分调用格式如下：

`int(f)`: 传回 f 对预设独立变数的积分值。

`int(f, 't')`: 传回 f 对独立变数 t 的积分值。

`int(f, a, b)`: 传回 f 对预设独立变数 t 的积分值, 积分区间为 $[a, b]$, a 和 b 为数值式。

`int(f, 't', a, b)`: 传回 f 对独立变数 t 的积分值, 积分区间为 $[a, b]$, a 和 b 为数值式。

`int(f, 'm', 'n')`: 传回 f 对预设变数 t 的积分值, 积分区间为 $[m, n]$, m 和 n 为符号式。

【例 3-24】利用 `int` 函数实现符号积分。

```
>> int(sym('log(x)/exp(x^2)'))
Warning: Explicit integral could not be found.
> In sym.int at 64
ans =
int(log(x)/exp(x^2), x)
>> f=sym('cos(x+2*y*t)')
f =
cos(x+2*y*t)
>> int(f)
ans =
sin(x + 2*t*y)
>> int(f,'s')
ans =
s*cos(x + 2*t*y)
>> int(f,pi/3,pi^3)
ans =
sin(2*t*y + 8727491006471547/281474976710656) - sin(1/3*pi + 2*t*y)
```

正如函数 `diff` 一样, 积分函数 `int` 对符号数组的每一个元素进行运算。

【例 3-25】符号数组积分。

```
>> F=sym(['a*2*x,+b^2*x;c*x^4,d*x;5,5*x'])
F =
[ a*2*x, +b^2*x]
[ c*x^4,   d*x]
[    5,   5*x]
>> diff(F)
ans =
[    2*a, b^2]
[ 4*c*x^3, d]
[    0,   5]
```

3.5.5 符号积分变换

傅里叶变换、变换和 Z 变换在许多研究领域都有着十分重要的作用, 特别是在信号处理和系统动态特性的研究中。傅里叶变换常常应用于连续系统, 而 FFT (快速傅里叶变换) 应用于离散系统; 拉普拉斯变换常用于连续系统 (微分方程), 其离散模式的 Z 变换则常常应用于离散系统 (差分方程)。

为此, MATLAB 提供了相应的函数来进行这些变换, 下面将重点讨论这些变换的具体使用方法。

1. 傅里叶变换及其逆变换

时域中的 $f(x)$ 与它的频域中的傅里叶变换存在如下关系:

$$f = f(x) \Rightarrow F = F(w) = \int_{-\infty}^{+\infty} f(x)e^{-iwx} dx$$

$$f(x) = \frac{1}{2} \int_{-\infty}^{+\infty} F(w)e^{iwx} dw$$

在 MATLAB 中分别由命令函数来完成此类变换，它们分别是 `fourier` 和 `ifourier`。对于傅里叶变换，其调用格式如下：

F = fourier(f): 对符号单值函数 f 中的默认变量 x （由 `findsym` 函数确定）计算傅里叶变换形式。默认的输出结果 F 是变量 w 的函数：

$$f(x) \Rightarrow F = F(w) = \int_{-\infty}^{+\infty} f(x)e^{-iwx} dx$$

若 $f=f(w)$ ，则 `fourier(f)` 返回变量为 t 的函数： $F=F(t)$ 。

F = fourier(f,v): 对符号单值函数 f 中的指定变量 v 计算傅里叶变换形式：

$$f = f(\oplus) \Rightarrow F = F(v) = \int_{-\infty}^{+\infty} f(x)e^{-i\oplus x} dx$$

F = fourier(f,u,v): 令符号函数 f 为变量 u 的函数，而 F 为变量 v 的函数：

$$f = f(\oplus) \Rightarrow F = F(v) = \int_{-\infty}^{+\infty} f(u)e^{-i\oplus u} du$$

【例 3-26】符号函数的傅里叶变换。

```
>> syms x w u v
>> f=sin(x)*exp(x^3)
f =
exp(x^3)*sin(x)
>> F=fourier(f)
F =
transform::fourier(exp(x^3)*sin(x), x, -w)
>> f1=x*exp(-abs(x))
f1 =
x/exp(abs(x))
>> F1=fourier(f1,u)
F1 =
-(4*i*u)/(u^2 + 1)^2
>> syms x real
>> f2=exp(-x^3*abs(v))*cos(v)/v
f2 =
cos(v)/(v*exp(x^3*abs(v)))
>> F2=fourier(f2,v,u)
F2 =
piecewise([0 < x, - i*atan((u - 1)/x^3) - i*atan((u + 1)/x^3)])
```

傅里叶逆变换使用 `ifourier` 函数来完成，其调用格式如下：

f = ifourier(F): 输出参量 $f=f(x)$ 为默认变量 w 的标量符号对象 F 的逆傅里叶积分变换，即 $F=F(w) \rightarrow f=f(x)$ 。若 $F=F(x)$ ，`ifourier(F)` 返回变量 t 的函数，即 $F=F(x) \rightarrow f=f(t)$ 。逆傅里叶积分变换定义如下：

$$f(x) = \frac{1}{2} \int_{-\infty}^{+\infty} F(w) e^{iwx} dw$$

$f = \text{ifourier}(F,u)$: 使函数 f 为变量 u (u 为标量符号对象) 的函数:

$$f(u) = \frac{1}{2} \int_{-\infty}^{+\infty} F(w) e^{i w u} dw$$

$f = \text{ifourier}(F,v,u)$: 使 F 为变量 v 的函数, f 为变量 u 的函数:

$$f(u) = \frac{1}{2} \int_{-\infty}^{+\infty} F(v) e^{i v u} dv$$

【例 3-27】符号函数的傅里叶反变换。

```
>> syms x w u v
f=sin(x)*exp(x^3)
f =
exp(x^3)*sin(x)
>> F1=ifourier(f)
F1 =
transform::fourier(exp(x^3)*sin(x), x, t)/(2*pi)
>> f1=x*exp(-abs(x))
f1 =
x/exp(abs(x))
>> F2=ifourier(f,u)
F2 =
transform::fourier(exp(x^3)*sin(x), x, u)/(2*pi)
>> syms x real
f2=exp(-x^3*abs(v))*cos(v)/v
f2 =
cos(v)/(v*exp(x^3*abs(v)))
>> F3=ifourier(f,v,u)
F3 =
exp(x^3)*dirac(u)*sin(x)
```

2. Z 变换及其逆变换

函数 f 的 Z 变换定义如下:

$$F(z) = \sum_{n=0}^{\infty} \frac{f(n)}{z^n}$$

在 MATLAB 中使用命令 `ztrans`, 其调用格式如下:

$F = \text{ztrans}(f)$: 对默认自变量为 n (就像由函数 `findsym` 确定的一样) 的单值函数 f 计算 Z 变换。输出参量 F 为变量 z 的函数: $f=f(n) \rightarrow F=F(z)$ 。若函数 $f=f(z)$, 则 `ztrans(f)` 返回一个变量为 w 的函数: $f=f(z) \rightarrow F=F(w)$ 。

$F = \text{ztrans}(f,w)$: 用符号变量 w 代替默认的 z 作为函数 F 的自变量:

$$F(w) = \sum_{n=0}^{\infty} \frac{f(n)}{w^n}$$

$F = \text{ztrans}(f,k,w)$: 对函数 f 指定的符号变量 k 计算 Z 变换:

$$F(w) = \sum_{n=0}^{\infty} \frac{f(k)}{w^n}$$

逆 Z 变换定义为: $f(n) = \frac{1}{2\pi i} \oint_{|z|=R} F(z)z^{n-1}dz$, $n=1,2,\dots$ 。最常见的三种 Z 反变换具体计算方法有幂级展开法、部分分式展开法和围线积分法。MATLAB 中采用围线积分法设计了求取 Z 反变换的 iztrans 函数。

【例 3-28】符号函数的 Z 变换及其反变换。

```
>> syms a n w
>> f=cos(a^x)+exp(x*w)
f =
cos(a^x) + exp(w*x)
>> z1=ztrans(f,w)
z1 =
w/(w - exp(w)) + ztrans(cos(a^x), x, w)
>> z2=iztrans(z1)
z2 =
(w*kronckerDelta(n, 0))/(w - exp(w)) + iztrans(ztrans(cos(a^x), x, w), x, n)
```

3. 拉普拉斯变换及逆变换

拉普拉斯变换定义为: $L(s) = \int_0^{+\infty} F(t)e^{-st}dt$, 在 MATLAB 中使用 laplace 函数来直接进行变换, 其调用格式如下:

laplace(F): 输出参量 $L=L(s)$ 为有默认符号自变量 t 的标量符号对象 F 的拉普拉斯变换。即 $F=F(t) \rightarrow L=L(s)$ 。若 $F=F(s)$, 则 laplace (F) 返回变量为 t 的函数 L。

laplace(F,t): 使函数 L 为变量 t (t 为标量符号自变量) 的函数:

$$L(s) = \int_0^{+\infty} F(x)e^{-sx}dx$$

laplace(F,w,z): 使 L 为变量 z 的函数, F 为变量 w 的函数:

$$L(z) = \int_0^{+\infty} F(w)e^{-zw}dw$$

【例 3-29】符号函数的拉普拉斯变换。

```
>> syms t x
>> f=x^3+t^5
f =
t^5 + x^3
>> laplace(f)
syms a t x
f=exp(-a*t+x^2)
ans =
x^3/s + 120/s^6
f =
exp(x^2 - a*t)
>> laplace(f,x)
ans =
exp(x^2)/(a + x)
```

逆拉普拉斯变换定义如下:

$$F(t) = \int_{c-i\infty}^{c+i\infty} L(s)e^{st} ds$$

在 MATLAB 中提供了 ilaplace 函数实现逆拉普拉斯变换。其调用格式如下：

F = ilaplace(L): 输出参量 $F=F(t)$ 为默认变量 s 的标量符号对象 L 的逆拉普拉斯变换。即 $F=F(w) \rightarrow f=f(t)$ 。若 $L=L(t)$ ，则 ifourier(L) 返回变量为 x 的函数 F 。即 $F=F(x) \rightarrow f=f(t)$ 。逆拉普拉斯变换定义为： $L(s) = \int_{c-i\infty}^{c+i\infty} F(t)e^{-st} dt$ ，其中 c 为使函数 $F(t)$ 的所有的奇点位于直线 $s=c$ 。

F = ilaplace(L,y): 使函数 F 为变量 y (y 为标量符号对象) 的函数：

$$F(y) = \int_{c-i\infty}^{c+i\infty} L(y)e^{sy} ds$$

F = ilaplace(L,y,x): 使函数 F 为变量 x 的函数， L 为变量 y 的函数：

$$F(x) = \int_{c-i\infty}^{c+i\infty} L(y)e^{xy} dy$$

【例 3-30】符号函数的拉普拉斯逆变换。

```
>> syms t x
f=x^3+t^5
f =
t^5 + x^3
>> ilaplace(f)
ans =
t^5*dirac(t) + dirac(t, 3)
>> syms a t x
f=exp(-a*t+x^2)
f =
exp(x^2 - a*t)
>> ilaplace(f)
ans =
(heaviside(t)*ilaplace(exp(x^2), x, t))/exp(a*t)
```

3.6 符号函数

函数是整个数学大厦的基础之一，对函数的描述以及操作也是 MATLAB 符号运算的内容之一。在 MATLAB 中提供了对函数相应的符号操作，包括复合函数、反函数以及符号函数的图形显示等。

3.6.1 复合函数的运算

若函数 $z=z(y)$ 的自变量 y 又是 x 的函数 $y=y(x)$ ，则求 z 对 x 的函数的过程称为复合函数运算过程。在 MATLAB 中，此过程可由功能复合函数 compose 来实现。

compose 函数的调用格式如下：

compose(f, g): 返回当 $f=f(x)$ 和 $g=g(y)$ 时的复合函数 $f(g(y))$ ，这里 x 为自定义的函数 f 的符号变量， y 为自定义的函数 g 的符号变量。

compose(f, g, z): 返回自变量为 z 的复合函数。

`compose(f, g, x, z)`: 返回复合函数 $f(g(z))$ 且使得 x 为 f 的独立变量。也就是说如果 $f=\cos(x/t)$, 是 `compose(f, g, x, y, z)`, 而 `compose(f,g,t,z)` 返回 $\cos(x/g(z))$ 。

`compose(f,g,x, y, z)`: 返回复合函数 $f(g(z))$ 并使得 x 为 f 的独立变量, y 为 g 的独立变量。即 $f=\cos(x/t)$, $g=\sin(y/u)$, `compose(f, g, x, y, z)` 返回 $\cos(\sin(z/u)/t)$, 而 `compose(f, g, x, u, z)` 返回 $\cos(\sin(y/z)/t)$ 。

【例 3-31】复合函数的运算示例。

```
>> syms x y z t u;
>> f=1/(1+x^3-y^3);
>> g=cos(y*x);
>> h=x*t;
>> p=exp(-y/u+t)
p =
exp(t - y/u)
>> compose(f,g)
ans =
1/(cos(x*y)^3 - y^3 + 1)
>> compose(f,g,t)
ans =
1/(cos(t*y)^3 - y^3 + 1)
>> compose(h,g,x,z)
ans =
t*cos(y*z)
>> compose(h,g,t,z)
ans =
x*cos(y*z)
>> compose(h,p,t,u,z)
ans =
x*exp(t - y/z)
```

3.6.2 反函数的运算

反函数运算也是符号函数中比较重要的一部分, 在 MATLAB 中提供 `finverse` 函数实现。其调用格式如下:

`g = finverse(f)`: g 为符号函数 f 的反函数。 f 为一符号函数表达式, 单变量为 ' x '。则函数 g 为一符号函数使得 $g(f(x))=x$ 。

`g = finverse(f,v)`: 返回的符号函数表达式的自变量为 v , 这里 v 为一符号, 是表达式的向量变量, 则 g 的表达式要使得 $g(f(v))=v$ 。当 f 包括不只一个变量时, 最好使用此型。

【例 3-32】反函数的运算示例。

```
>> syms x y a
>> f=x^3+y^3+2*a;
>> finverse(f,y)
Warning: finverse(x^3 + y^3 + 2*a) is not unique.
> In sym.finverse at 46
ans =
(y - 2*a - x^3)^(1/3)
>> finverse(f)
```


Warning: finverse(x^3 + y^3 + 2*a) is not unique.

> In sym.finverse at 46

ans =

(x - 2*a - y^3)^(1/3)

此时由于没有指明自变量，MATLAB 给出警告信息，且以默认变量 x 给出结果。

3.6.3 符号函数的可视化

为了把符号运算得到的数值结果用图形显示出来，MATLAB 符号工具箱提供了许多符号作图函数来实现符号函数的可视化。

符号绘图函数可分为二维绘图和三维绘图，主要的绘图函数见表 3-1。

表 3-1 常见符号绘图函数

函数名	功能说明
ezplot	绘制符号表达式的自变量与对应的函数值的曲线
ezplot3	绘制符号表达式的自变量与对应的函数值的三维曲线
ezcontour	绘制等高线
ezcontourf	绘制带填充颜色的等高线
ezmesh	绘制三维网线图
ezmeshc	绘制等高线的三维网线图
ezpolar	绘制极坐标
ezsurf	绘制三维曲面图
ezsurfz	绘制带三维等高线的三维曲面图
fplot	绘制误差图

下面对几个符号绘图函数展开应用。

1. ezcontourf 函数

ezcontourf 函数的调用格式如下：

ezcontourf(fun): 绘制二元符号函数 $f=f(x,y)$ 的等高线图，在不同的等高线之间填充不同颜色，函数默认的平面区域为 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 。

ezcontourf(fun,domain): 函数在指定的定义域 domain 中绘制等高线图。

ezcontourf(...,n): 函数在指定的 $N \times N$ 的栅格点内绘制等高线图，N 的默认值为 60。

ezcontourf(axes_handle,...): 在指定的坐标轴 axes_handle 里面绘制等高线图。

h = ezcontourf(...): 返回等高线的图像对象的句柄值。

ezcontourf 函数自动添加标题和坐标轴标签。

【例 3-33】运用函数 ezcontourf 绘制下面函数的等高线图。

$$f(x, y) = 3(1-x)^2 e^{-x^2} - (y+1)^2 - 10 \left(\frac{x}{5} - x^3 - y^5 \right) e^{-x^2 - y^2} - \frac{1}{3} e^{-(x+1)^2 - y^2}$$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
```

```
f = ['3*(1-x)^2*exp(-(x^2)-(y+1)^2)',...
```

```
'- 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2)',...
```

```
'- 1/3*exp(-(x+1)^2 - y^2)'];
ezcontourf(f,[-3,3],50)
```

运行程序，效果如图 3-1 所示。

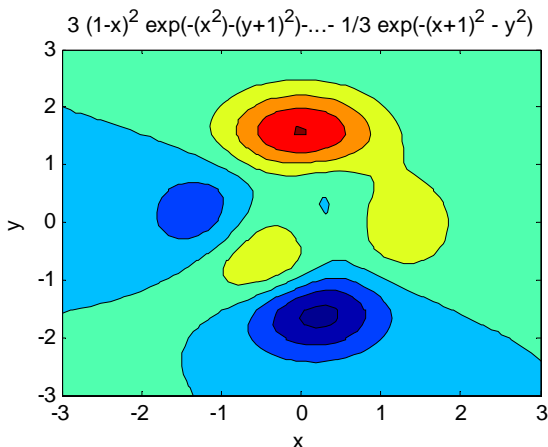


图 3-1 绘制不同颜色的等高线图

2. ezmeshc 函数

其的调用格式如下：

ezmeshc(fun): 绘制二元符号函数 $z=f(x,y)$ 的网格图，且同时在 xy 平面内显示其等高线图，函数默认的平面区域为 $[-2\pi < x < 2\pi, -2\pi < y < 2\pi]$ 。

ezmeshc(fun,domain): 函数在指定的定义域 domain 中绘制网格图，domain 可以是四维向量 $[xmin, xmax, ymin, ymax]$ 或者二维向量 $[a, b]$ 。

ezmeshc(funx,funy,funz): 函数在指定的矩形定义域范围 $[-2\pi < s < 2\pi, -2\pi < t < 2\pi]$ 中绘制参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 和 $z=z(s,t)$ 的二元函数 $z=f(x,y)$ 网格图。

ezmeshc(funx,funy,funz,[smin,smax,tmin,tmax]): 函数在指定的矩形定义域范围 $[smin < s < smax, tmin < t < tmax]$ 中绘制参数形式函数 $x=x(s,t)$ 、 $y=y(s,t)$ 和 $z=z(s,t)$ 的二元函数 $z=f(x,y)$ 网格图。

ezmeshc(funx,funy,funz,[min,max]): 函数在指定的定义域 $[min, tmax]$ 绘制二元函数 $z=f(x,y)$ 网格图。

ezmeshc(...,n): 函数在指定的 $N \times N$ 的栅格点内绘制网格图， N 的默认值为 60。

ezmeshc(...,'circ'): 函数在一圆形区域内绘制网格图。

ezmesh(axes_handle,...): 函数在指定的坐标轴 axes_handle 里面绘制网格图。

h = ezmeshc(...): 函数返回网格图的图像对象的句柄值。

【例 3-34】 运用函数 ezmeshc 绘制下面函数的网格图。

$$z = x * k - z * k - 1$$

其实现的 MATLAB 程序代码如下：

```
% 首先定义 li3_34 函数
function z = li3_34 (x,y,k)
z = x.^k - y.^k - 1;
%
```

然后在命令窗口中输入以下代码：

```
>> ezmeshc(@ (x,y) li3_34 (x,y,2))
```

运行程序，效果如图 3-2 所示。

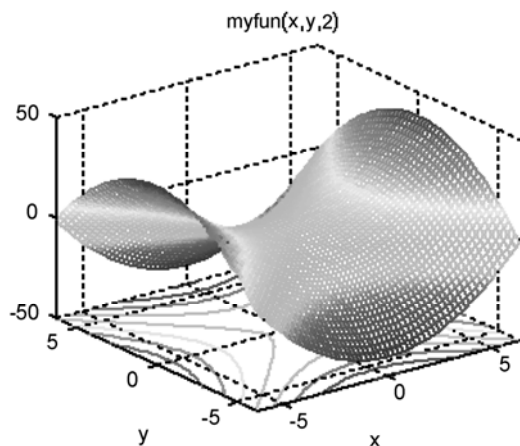


图 3-2 绘制带等高线的三维网线图

3. fplot 函数

其调用格式如下：

fplot(fun,limits): 绘制字符串 **fun** 指定函数名的函数在 **x** 轴区间为 **limits=[xmin xmax]** 的函数图。若 **limits=[xmin xmax ymin ymax]**，则 **y** 轴也被输入限制的区间。**fun** 必须为一 **M** 文件的函数的函数名或对变量 **x** 的可执行字符串，此字符串被送入运行字符串命令函数 **eval** 后被执行。函数 **fun(x)** 必须要返回一针对向量 **x** 的每一元素的结果行向量。

fplot(fun,limits,LineStyle): 其中用 **LineStyle** 用来指定线条类型。

fplot(fun,limits,tol): 其中 **tol** < 1 用来指定相对误差精度。默认值为 **tol=2e-3**。

fplot(fun,limits,n): 其中 **n** >= 1，指定以最少 **n+1** 个点来绘制函数图。默认 **n=1**。最大步长被约束在不小于 $(1/n) \cdot (x_{\max} - x_{\min})$ 。

fplot(fun,lims,...): 只返回应用来绘图点的向量值，而不绘制图形。用户可自己用 **plot(x,y)** 来输出图形。

【例 3-35】利用 **fplot** 函数来绘制图形。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
subplot(2,2,1);
fplot('humps',[0 1]);
subplot(2,2,2);
fplot('abs(exp(-j*x*(0:9))*ones(10,1))',[0 2*pi]);
subplot(2,2,3);
fplot('tan(x),sin(x),cos(x)',2*pi*[-1 1 -1 1]);
subplot(2,2,4);
fplot('sin(1./x)',[0.01 0.1],1e-3);
```

运行程序，效果如图 3-3 所示。

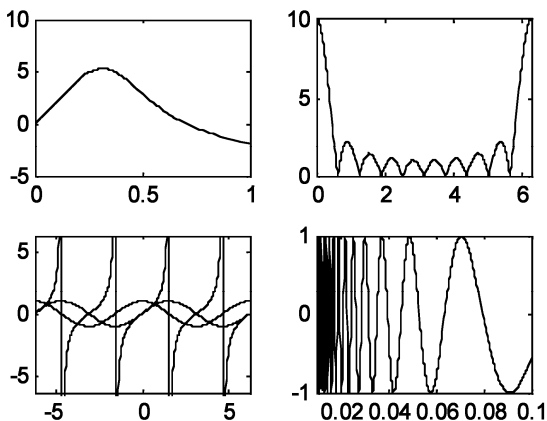


图 3-3 4 种不同的调用格式绘制形式

其中: `humps.m` 为 MATLAB 内部函数, 用来进行绘图、积分等演示, 具体结构如下:

```
function [out1,out2]=humps(x)
if nargin==0,
    x=0:0.05:1;
end
y=1./((x-.3).^2+0.01)+1./((x-0.9).^2+0.04)-6;
```

4. `ezplot` 函数

其调用格式如下:

`ezplot(fun)`: 绘制显示符号函数 $y=f(x)$ 在范围 $[-\pi < x < \pi]$ 上的函数 $f(x)$ 的图形, 隐式符号函数 $y=f(x,y)$ 在范围 $[-2\pi < x < -2\pi, -2\pi < y < 2\pi]$ 上的函数 $f(x,y)$ 的图形。

`ezplot(fun,[min,max])`: 函数在指定的定义域 $[min, max]$ 中绘制图形。

`ezplot(fun2)`: 绘制函数 $fun2(x, y)=0$ 在默认定义域 $[-2\pi < x < -2\pi, -2\pi < y < 2\pi]$ 上的图形。

`ezplot(fun2,[xmin,xmax,ymin,ymax])`: 绘制函数 $fun2(x, y)=0$ 在指定定义域 $[xmin < x < xmax, ymin < y < ymax]$ 上的图形。

`ezplot(fun2,[min,max])`: 绘制函数 $fun2(x,y)=0$ 在指定定义域 $[min < x < max, min < y < max]$ 上的图形。

`ezplot(funx,funy)`: 绘制由平面曲线 $funx(t)$, $funy(t)$ 定义的参数在默认定义域 $[0 < t < 2\pi]$ 上的图形。

`ezplot(funx,funy,[tmin,tmax])`: 在定义域 $[tmin, tmax]$ 内绘制参数函数 $funx(t)$ 、 $funy(t)$ 的图形。

`ezplot(...,figure_handle)`: 在由 `figure_handle` 定义的图形窗口内绘制出给定函数在特定定义域内的图形。

`ezplot(axes_handle,...)`: 在指定的坐标轴 `axes_handle` 绘制图形。

`h = ezplot(...)`: 返回曲线对象的图形句柄值。

【例 3-36】运用函数 `ezmeshc` 绘制下面函数的网格图。

$$z=x^*k-z^*k-1$$

其实现的 MATLAB 程序代码如下:

```
% 首先定义 li3_36 函数
function z = li3_36 (x,y,k)
z = x.^k - y.^k - 1;
```

然后在命令窗口中输入以下代码：

```
>> ezplot(@(x,y) li3_36(x,y,2))
```

运行程序，效果如图 3-4 所示。

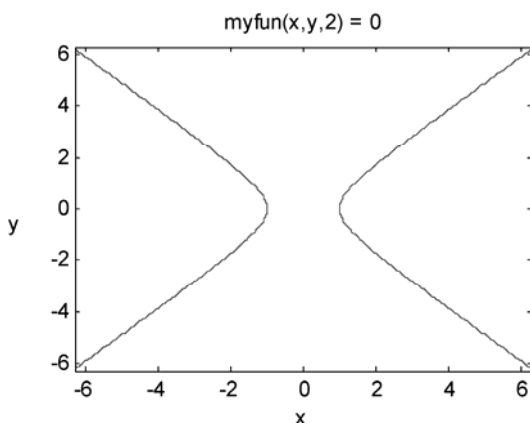


图 3-4 函数 ezplot 绘制的极坐标图形

3.7 符号方程的求解

3.7.1 代数方程的求解

1. 线性方程组的符号解

矩阵运算式是求解线性方程组最简便有效的方法。无论数据对象是数值还是符号，实现矩阵运算的命令形式几乎完全相同。因此，对于求解线性方程组符号解的问题，可以套用求数值解的方法进行。

【例 3-37】求下面线性方程组的解：

$$\begin{cases} d + \frac{n}{2} + \frac{p}{2} - q = 0 \\ n + d + q - p = 10 \\ q + d - \frac{n}{4} - p = 0 \\ q + p - n - 8d = 1 \end{cases}$$

该方程组的矩阵形式是：

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -\frac{1}{4} & -1 & 1 \\ -8 & -1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} d \\ n \\ p \\ q \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \\ 0 \\ 0 \end{bmatrix}, \text{ 该式简记为 } \mathbf{AX}=\mathbf{b}.$$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
```

```
A=sym([1 1/2 1/2 -1;1 1 -1 1;1 -1/4 -1 1;-8 -1 1 1]);
b=sym([0;10;0;1]);
X=A\b
```

运行程序，输出如下：

```
X =
    1
    8
    8
    9
```

2. 一般代数方程组的解

除了上面所讲的线性 (Linear) 方程组的解以外，更一般的代数方程包括非线性 (Nonlinear) 和超越方程 (Transcendental equation) 等，求解函数是 solve。当方程组不存在符号解又无其他自由参数时，solve 将给出数值解。该函数的调用格式如下：

solve(eq): 输入参数 eq 是符号表达式或字符串。若 eq 是一符号表达式 x^2-2x-1 或一个没有等号的字符串 “ x^2-2x-1 ”，则 solve(eq) 对方程 eq 中的默认变量（由函数 findsym(eq) 确定的变量）求解方程 $eq=0$ 。

solve(eq,var): 对符号表达式或没有等号的字符串 eq 指定的变量 var 求解方程 $eq(var)=0$ 。

solve(eq1,eq2,...,eqn): 输入参数 eq1, eq2, ..., eqn 可以是符号表达式或字符串。该命令对方程组 eq1, eq2, ..., eqn 中函数 findsym 确定的 n 个变量如 x1, x2, ..., xn 求解。

g = solve(eq1,eq2,...,eqn,var1,var2,...,varn): 对方程组 eq1, eq2, ..., eqn 中指定的 n 个变量，如 var1, var2, ..., varn 求解。

【例 3-38】代数方程的求解。

```
>> solve('a*x^2 + b*x + c')
ans =
  -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
  -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
>> solve('a*x^2 + b*x + c','b')
ans =
  -(a*x^2 + c)/x
>> S = solve('x + y = 1','x - 11*y = 5')
S =
    x: [1x1 sym]
    y: [1x1 sym]
>> A = solve('a*u^2 + v^2', 'u - v = 1', 'a^2 - 5*a + 6')
A =
    a: [4x1 sym]
    u: [4x1 sym]
    v: [4x1 sym]
>> A.a
ans =
    3
    2
    2
    3
>> A.u
```

```
ans =
(3^(1/2)*i)/4 + 1/4
(2^(1/2)*i)/3 + 1/3
1/3 - (2^(1/2)*i)/3
1/4 - (3^(1/2)*i)/4
>> A.v
ans =
(3^(1/2)*i)/4 - 3/4
(2^(1/2)*i)/3 - 2/3
- (2^(1/2)*i)/3 - 2/3
- (3^(1/2)*i)/4 - 3/4
```

【例 3-39】小军想去买学习用具，他从小钱罐倒出自己存的硬币并清点，然后他发现：

- (1) 1 角的硬币数加上 1 分和 5 分的硬币总数的 $1/2$ 等于 5 角的硬币数。
- (2) 1 分的硬币数比 5 分、1 角以及 5 角的硬币总数多 10。
- (3) 5 角的硬币数比 5 分、1 角以及 5 角的硬币总数多 10。
- (4) 5 角的硬币数和 1 分的硬币数比 5 分的硬币数加上 8 倍的 1 角的硬币数多 1。

如果买书为 4.00 元，买笔花 1.5 元，橡皮擦为 5 角，他有足够的钱去买这三样东西吗？

首先，根据以上给出的信息列出一组线性方程，假如 p , n , d 和 q 分别表示 1 分、5 分、1 角和 5 角的硬币数。建立方程组如下：

$$\begin{cases} d + \frac{n+p}{2} - q = 0 \\ n + d + q - p = 10 \\ q + d - p - \frac{n}{4} = 0 \\ q + p - n - 8d = -1 \end{cases}$$

其实现的 MATLAB 程序代码如下：

```
>> equ1='d+(n+p)/2=q';
>> equ2='p=n+d+q-10';
>> equ3='q+d=p+n/4';
>> equ4='q+p=n+8*d-1';
>> [pennies,nickles,dimes,aquarters]=solve(equ1,equ2,equ3,equ4,'p,n,d,q')
```

运行程序，输出如下：

```
pennies =
3
nickles =
8
dimes =
16
aquarters =
15
```

所以，得出小军有 3 枚 1 分的硬币，8 枚 5 分的硬币，16 枚 1 角的硬币，15 枚 5 角的硬币。

```
>> money=0.01*pennies+0.05*nickles+0.1*dimes+0.5*aquarters
money =
953/100
```

这就意味着他有足够的钱去买书、笔和橡皮擦，还剩 2.36 元。

3.7.2 微分方程的求解

1. 单个微分方程

常微分方程有时候很难求解，MATLAB 提供了强大的工具，可以帮助求解微分方程。函数 `dsolve` 计算常微分方程的符号解。因为要求解微分方程，就需要用一种方法将微分包含在表达式中。所以，`dsolve` 句法与大多数其他函数有一些不同，用字母 `D` 来表示求微分，`D2`、`D3` 等表示重复求微分，并以此来设定方程。任何 `D` 后所跟的字母为因变量。方程 $\frac{d^2y}{dx^2}=0$ 用符号表达式 `D2y=0` 来表示。独立变量可以指定或由 `symvar` 规则选定为默认。例如，一阶方程 $\frac{dy}{dx}=1+y^2$ 的通解如下：

```
>> dsolve('Dy=1+y^2')    %求通解
ans =
tan(t+C1)
```

其中：`C1` 是积分常数。求解初值 $y(0)=1$ 的同一个方程就可产生：

```
>> dsolve('Dy=1+y^2','y(0)=1')    %增加一个初始条件
ans =
tan(t+1/4*pi)
```

独立变量可用如下形式指定：

```
>> dsolve('Dy=1+y^2','y(0)=1','v')    %求 dy/dv 的解
ans =
tan(v+1/4*pi)
```

【例 3-40】有一个二阶微分方程如下，该方程有两个初始条件：

$$\frac{d^2y}{dx^2} = \cos(2x) - y, \quad \frac{dy}{dx}(0) = 0, \quad y(0)=1$$

```
>> y=dsolve('D2y=cos(2*x)-y','Dy(0)=0','y(0)=1')
y =
cos(t)*(1-cos(2*x))+cos(2*x)
>> y=simple(y)    %可以化简类似于此的 y 方程
y =
cos(t)*(1-cos(2*x))+cos(2*x)
```

通常，要求解的微分方程含有一阶以上的项，并以下述的形式表示：

$$\frac{d^2y}{dx^2} - 2\frac{dy}{dx} - 3y = 0f$$

通解如下：

```
>> y=dsolve('D2y-2*Dy-3*y=0')
y =
C1*exp(3*t)+C2*exp(-t)
```

加上初始条件 $y(0)=0$ 和 $y(1)=1$ 可得到：

```
>> y=dsolve('D2y-2*Dy-3*y=0','y(0)=0','y(1)=1','x')
y =
1/(exp(3)-exp(-1))*exp(3*x)-1/(exp(3)-exp(-1))*exp(-x)
>> y=simple(y)    % 该式类似一个简化式
y =
(exp(3*x)-exp(-x))/(exp(3)-exp(-1))
```



```
>> pretty(y)
```

$$\frac{\exp(3x) - \exp(-x)}{\exp(3) - \exp(-1)}$$

现在来绘制感兴趣的区域内的结果，效果如图 3-5 所示。

```
>> ezplot(y,[-6 2])
```

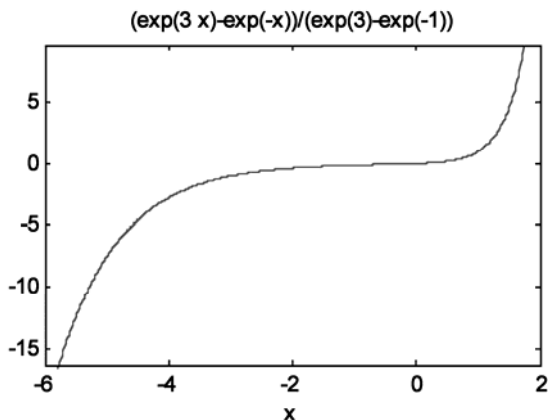


图 3-5 符号函数图

2. 求解微分方程组

函数 `dsolve` 也可同时处理若干个微分方程式。

【例 3-41】设有如下两个线性一阶方程。

$$\frac{du}{dx} = 6u + 3v, \quad \frac{dv}{dx} = -3u + 9v$$

求其通解。

其实现的 MATLAB 程序代码如下：

```
>> [u,v]=dsolve('Du=6*u+3*v','Dv=3*u+9*v')
u =
(5^(1/2)*C10*exp((15*t)/2 + (3*5^(1/2)*t)/2))/2 - (C11*exp((15*t)/2 - (3*5^(1/2)*t)/2))/2 -
(C10*exp((15*t)/2 + (3*5^(1/2)*t)/2))/2 - (5^(1/2)*C11*exp((15*t)/2 - (3*5^(1/2)*t)/2))/2
v =
C10*exp((15*t)/2 + (3*5^(1/2)*t)/2) + C11*exp((15*t)/2 - (3*5^(1/2)*t)/2)
```

加上初始条件： $u(0)=0$ ， $v(0)=0$ ，可得到：

```
>> [u,v]=dsolve('Du=6*u+3*v','Dv=3*u+9*v','u(0)=0','v(0)=1')
u =
(exp((15*t)/2 + (3*5^(1/2)*t)/2)*(5^(1/2) + 1))/4 - (exp((15*t)/2 - (3*5^(1/2)*t)/2)*(5^(1/2) - 1))/4 -
(5^(1/2)*exp((15*t)/2 - (3*5^(1/2)*t)/2)*(5^(1/2) - 1))/20 - (5^(1/2)*exp((15*t)/2 +
(3*5^(1/2)*t)/2)*(5^(1/2) + 1))/20
v =
(5^(1/2)*exp((15*t)/2 - (3*5^(1/2)*t)/2)*(5^(1/2) - 1))/10 + (5^(1/2)*exp((15*t)/2 +
(3*5^(1/2)*t)/2)*(5^(1/2) + 1))/10
```

第 4 章 数据分析与概率分布

不管是计算机编程，还是网络设计、分析实际问题，随机数都有广泛的应用。

4.1 随机数的产生

本节主要介绍产生不同概率分布的随机数及其使用方法。MATLAB 为用户提供了多种随机函数，它们大多数存放在自带的统计工具箱之中，比如正态分布、Beta 分布和泊松分布等。统计工具箱被放置在路径 MATLAB6p5\toolbox\stats 中，用户可以进入这个路径查阅更多的信息。

4.1.1 一般随机数生成

在 MATLAB 中使用随机数，有时希望每次开启 MATLAB 之后，运行随机数的程序都得到同一个结果。建议大家首先设置一下随机函数的状态，即设定 state 的取值（MATLAB 5.0 版本之前称为 seed，之后的版本同样可以使用），其具体格式如下：

```
rand('state', m);
```

```
randn('state', n);
```

这里 m 和 n 是随机函数的状态，它们可以是一个任意的数值，也可以是 double 型数据，比如 1、2、3 等，还可以依据当前时钟数值取值，比如 sum(100*clock)。在实际应用中希望含随机数的程序每次开启 MATLAB 的运行结果都一样，所以应设置 m 和 n 为定值。

在实际应用中，经常使用的随机函数是 rand，它产生 0~1 之间均匀分布的随机数。扩展到在任意两个数值之间产生均匀分布随机数的方法可以采用下面的 MATLAB 程序代码：

```
>> a=30;  
b=20;  
rand('state',15);    %设定状态为 15  
x=a*(a-b)*rand(6)    %产生 a 和 b 之间的随机数
```

运行程序，输出如下：

```
x =  
165.3134 128.6269 237.1007 276.2902 208.5287 77.0500  
95.5123 32.6279 0.6567 208.5202 87.9032 72.3723  
176.6579 221.3279 219.1274 25.1766 275.8056 78.2214  
72.3313 150.5879 62.8162 133.2352 76.5878 221.5212  
240.4655 198.6717 130.6522 52.4988 273.1404 22.9584  
271.9944 234.0947 89.1607 39.7182 286.8763 96.6092
```

这里 a 和 b 取值不等即可，没有大小关系限制。rand 函数的调用格式如下：

```
r = rand(n)  
rand(m,n)  
rand([m,n])  
rand(m,n,p,...)
```

```
rand([m,n,p,...])
rand
rand(size(A))
r = rand(..., 'double')
r = rand(..., 'single')
```

它们依次产生 0 维（单个数值）、1 维、2 维和高维数组。产生正态分布（或者称为高斯分布）的函数是 `randn`，它的使用方法类似于 `rand` 函数，这里不再展开介绍。在 MATLAB 中，`rand` 和 `randn` 函数是开发技术人员嵌入 MATLAB 的函数，与其他随机函数不放在一起。对于一般的用户而言，没有特别需要统计工具箱可以不用安装。

在统计工具箱中，`random` 可以产生多种分布的随机数。其调用格式如下：

```
R=random('name', A, M, N)
```

其中：`name` 是分布的名称；`A` 是该分布的参数；`M` 和 `N` 表示生成矩阵的大小。其中 `name` 参数可以输入 Beta, Binomial, Chisquare, Exponential, F, Gamma, Geometric, Hypergeometric, Lognormal, Negative Binomial, Noncentral F, Noncentral t, Noncentral Chi-square, Normal, Poisson, Rayleigh, T, Uniform, Discrete Uniform, Weibull。

从这些单词可以看出相应分布的名称。在使用时可以仔细阅读相关帮助文档。通过测试，可以使用语句“`rand('state', m)`”来控制 `random` 函数的状态，而“`randn('state', m)`”不能控制 `random` 的状态。在实际应用中，确定随机数的状态还可以使用下面的方法完成，即用函数 `save` 把当前的随机数保存下来，以后使用该随机数时用 `load` 函数读入即可，但是这样需要产生一个多余数据文件。

【例 4-1】`random` 函数生成随机数示例。

```
>> rn=random('Normal',0,1,2,4)
rn =
    -0.3999    0.8156    1.2902    1.1908
     0.6900    0.7119    0.6686   -1.2025
>> rp=random('Poisson',1:7,1,7)
rp =
     0     4     1     5     3     9    11
```

4.1.2 其他分布的随机函数

在 MATLAB 中还提供了一些专门随机函数，下面对这些随机函数展开介绍。

1. `betarnd` 函数

功能：beta 随机数生成器。其调用格式如下：

```
R = betarnd(A,B)
R = betarnd(A,B,v)
R = betarnd(A,B,m,n)
R = betarnd(A,B,m,n,o,...)
```

`betarnd` 函数相应分布的概率密度函数的数学公式如下：

$$f(x|a,b) = \frac{x^{a-1}(1-x)^{b-1}I_{0,1}(x)}{B(a,b)}$$

【例 4-2】betarnd 函数应用示例。

```
>> a = [1 1;2 2];
b = [1 2;1 2];
>> r = betarnd(a,b)
r =
    0.7876    0.1722
    0.1236    0.5608
>> r = betarnd(10,10,[1 5])
r =
    0.3711    0.5249    0.4757    0.4750    0.4466
>> r = betarnd(4,2,2,3)
r =
    0.7906    0.5242    0.9266
    0.4301    0.8299    0.7545
```

2. binornd 函数

功能：生成二项式随机数生成器。其调用格式如下：

$R = \text{binornd}(N,P)$

$R = \text{binornd}(N,P,v)$

$R = \text{binornd}(N,p,m,n)$

binornd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|n,p) = \binom{n}{x} p^x (1-p)^{n-x} I_{0,1,\dots,n}(x)$$

【例 4-3】二项式随机数生成器示例。

```
>> n = 10:10:60;
>> r1 = binornd(n,1./n)
r1 =
     1     1     1     1     1     2
>> r2 = binornd(n,1./n,[1 6])
r2 =
     1     1     1     1     0     0
>> r3 = binornd(n,1./n,1,6)
r3 =
     1     1     2     1     0     1
```

3. chi2rnd 函数

功能：卡方分布的随机数生成器。其调用格式如下：

$R = \text{chi2rnd}(V)$

$R = \text{chi2rnd}(V,u)$

$R = \text{chi2rnd}(V,m,n)$

chi2rnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|v) = \frac{x^{(v-2)/2} e^{-x/2}}{[2^{v/2} \Gamma(v/2)]}$$

【例 4-4】生成卡方分布的随机数生成器。

```
>> r = chi2rnd(1:6)
```

```
r =
    1.3777    1.0934    0.3004    4.0428    1.9079   11.4188
>> r = chi2rnd(6,[1 6])
r =
    3.1124    7.2533    6.0822    2.8535    0.9212    5.1424
>> r = chi2rnd(1:6,1,6)
r =
    0.2385    2.6095    3.6110    9.8909    6.3175    3.4954
```

4. exprnd 函数

功能：指数分布的随机数生成器。其调用格式如下：

`R = exprnd(mu)`

`R = exprnd(mu,v)`

`R = exprnd(mu,m,n)`

exprnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|\mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

【例 4-5】exprnd 函数应用示例。

```
>> n1 = exprnd(5:10)
n1 =
    7.9093    0.9105    1.9987    4.0647    0.6917   10.5552
>> n2 = exprnd(5:10,[1 6])
n2 =
    4.2718    4.1435   18.1966    2.6947   12.7703    1.4413
>> n3 = exprnd(5,2,3)
n3 =
    2.8897    3.4086    5.1025
    0.3133    3.7261    1.5478
```

【例 4-6】生成指数分布的随机数。

```
clc;
%设置指数分布的参数
mu=4;
%产生 len 个随机数
len=5;
y1=exprnd(mu, [1 len])
%产生 P 行 Q 列的矩阵
P=3;
Q=4;
y2=exprnd(mu, P,Q)
%显示指数分布的柱状图
M=1000;
y3=exprnd(mu, [1 M]);
figure(1);
t=0:0.2:max(y3);
hist(y3,t);
```

```
axis([0 max(y3) 0 100]);
xlabel('取值');
ylabel('计数值');
```

运行程序，输出如下，效果如图 4-1 所示。

```
y1 =
    1.0911    0.2101    0.3163    5.6231    5.2701
y2 =
    5.6328    0.5772    7.7739    2.5458
    2.5925    1.3091    2.3738    2.9560
    1.0329   12.9228    3.9302    1.1686
```

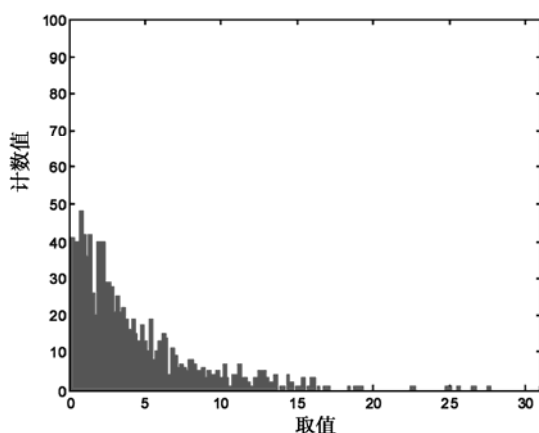


图 4-1 指数分布的柱状图

5. frnd 函数

功能：F 分布随机数生成器。其调用格式如下：

$R = \text{frnd}(V1, V2)$

$R = \text{frnd}(V1, V2, v)$

$R = \text{frnd}(V1, V2, m, n)$

frnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x | v_1, v_2) = \frac{\Gamma\left[\frac{v_1 + v_2}{2}\right]}{\Gamma\left[\frac{v_1}{2}\right]\Gamma\left[\frac{v_2}{2}\right]} \left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}} \frac{x^{\frac{v_1-2}{2}}}{\left[1 + \left(\frac{v_1}{v_2}\right)x\right]^{\frac{v_1+v_2}{2}}}$$

【例 4-7】F 分布随机数生成器示例。

```
>> n1 = frnd(1:6,1:6)
n1 =
    0.7930   15.6780    0.6768    0.5160    0.6316    0.5852
>> n2 = frnd(2,2,[2 3])
n2 =
    16.6857    1.6240    0.1508
     2.8106    0.4999    1.0808
>> n3 = frnd([1 2 3;4 5 6],1,2,3)
```

```
n3 =
    1.0e+003 *
    0.0067    0.0142    0.0593
    0.2446    8.7664    0.0008
```

6. gamrnd 函数

功能：gamma 随机数生成器。其调用格式如下：

$R = \text{gamrnd}(A,B)$

$R = \text{gamrnd}(A,B,v)$

$R = \text{gamrnd}(A,B,m,n)$

gamrnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

【例 4-8】gamma 随机数生成器示例。

```
>> n1 = gamrnd(1:5,6:10)
n1 =
    1.6247    7.3053   27.6699   19.1874   65.6580
>> n2 = gamrnd(5,10,[1 5])
n2 =
    60.0683   31.0712   41.1608   25.4232   13.4271
>> n3 = gamrnd(2:6,3,1,5)
n3 =
    9.8755    5.7187   12.9898   15.5723   17.1538
```

7. geornd 函数

功能：几何分布的随机数生成器。

$R = \text{geornd}(P)$

$R = \text{geornd}(P,v)$

$R = \text{geornd}(P,m,n)$

geornd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|p) = p(1-p)^x I_{(0,1,K)}(x)$$

【例 4-9】geornd 函数应用示例。

```
>> r1 = geornd(1 ./ 2.^(1:6))
r1 =
     0     2     2     8    82    33
>> r2 = geornd(0.01,[1 5])
r2 =
    101    28    58    87    50
>> r3 = geornd(0.5,1,6)
r3 =
     1     2     0     0     0     1
```

【例 4-10】生成几何分布的随机数。

```
clc;
%设置几何分布的参数
```

```

p=0.05;
%产生 len 个随机数
len=5;
y1=geornd(p, [1 len])
%产生 P 行 Q 列的矩阵
P=3;
Q=4;
y2=geornd(p, P,Q)
%显示几何分布的柱状图
M=1000;
y3=geornd(p, [1 M]);
figure(1);
t=0:2:max(y3);
hist(y3,t);
axis([0 max(y3) 0 100]);
xlabel('取值');
ylabel('计数值');

```

运行程序，输出如下，效果如图 4-2 所示。

```

y1 =
     8     0    17    71    32
y2 =
     7    27    13     4
     9     0    19    11
    48     1     6     7

```

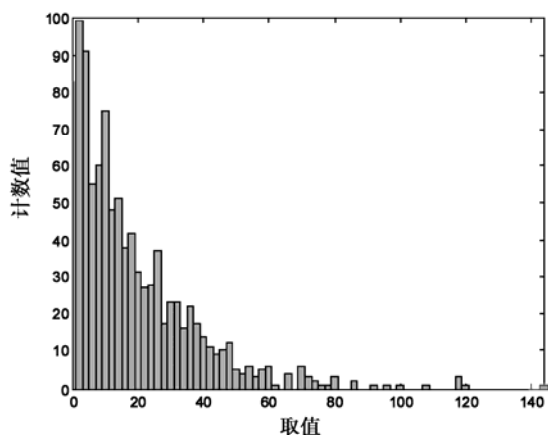


图 4-2 几何分布的柱状图

8. hygernd 函数

功能：超几何分布随机数生成器。其调用格式如下：

$R = \text{hygernd}(M, K, N)$

$R = \text{hygernd}(M, K, N, v)$

$R = \text{hygernd}(M, K, N, m, n)$

hygernd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|M,K,N) = \frac{\binom{K}{x} \binom{M-K}{N-x}}{\binom{M}{N}}$$

【例 4-11】hygernd 函数应用示例。

```
>> numbers = hygernd(1000,40,50)
numbers =
     2
```

【例 4-12】生成超几何分布的随机数。

```
clc;
%设置超几何分布的参数
M=1000;
K=50;
n=20;
%产生 len 个随机数
len=5;
y1=hygernd(M,K,n,[1 len])
%产生 P 行 Q 列的矩阵
P=3;
Q=4;
y2=hygernd(M,K,n,P,Q)
%显示超几何分布的柱状图
M=1000;
y3=hygernd(M,K,n,[1 M]);
figure(1);
t=0:1:max(y3);
hist(y3,t);
axis([0 max(y3) 0 500]);
xlabel('取值');
ylabel('计数值');
```

运行程序，输出如下，效果如图 4-3 所示。

```
y1 =
     3     2     0     2     2
y2 =
     3     2     2     2
     0     0     1     0
     1     0     0     2
```

9. lognrnd 函数

功能：对数正态分布随机数生成器。其调用格式如下：

R = lognrnd(mu,sigma)

R = lognrnd(mu,sigma,v)

R = lognrnd(mu,sigma,m,n)

lognrnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|\mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}$$

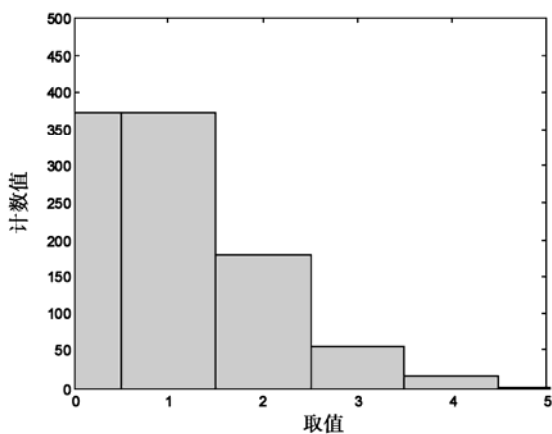


图 4-3 超几何分布的柱状图

【例 4-13】对数正态分布随机数生成器示例。

```
>> m = 1;
v = 2;
mu = log((m^2)/sqrt(v+m^2));
sigma = sqrt(log(v/(m^2)+1));
[M,V]= lognstat(mu,sigma)
M =
    1
V =
    2.0000
>> X = lognrnd(mu,sigma,1,1e6);
>> MX = mean(X)
MX =
    1.0020
>> VX = var(X)
VX =
    2.0024
```

10. nbinrnd 函数

功能：负二项分布随机数生成器。其调用格式如下：

RND = nbinrnd(R,P)

RND = nbinrnd(R,P,m)

RND = nbinrnd(R,P,m,n)

nbinrnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|r, p) = \binom{r+x-1}{x} p^r (1-p)^x I_{(0,1,\dots)}(x)$$

【例 4-14】负二项分布随机数生成器示例。

```
>> r = nbinrnd(3,0.01,1,6)+3
```

```
r =
```

```
269    113    171    181    223    170
```

11. ncfrnd 函数

功能：非中心 F 分布随机数生成器。其调用格式如下：

```
R = ncfrnd(NU1,NU2,DELTA)
```

```
R = ncfrnd(NU1,NU2,DELTA,v)
```

```
R = ncfrnd(NU1,NU2,DELTA,m,n)
```

ncfrnd 函数相应分布的概率密度函数的数学公式如下：

$$(x | v_1, v_2, \delta) = \sum_{k=0}^{\infty} \frac{e^{-\delta/2} (\delta/2)^k}{B(v_1/2, v_2/2 + k) k!} \left(\frac{v_1}{v_2} \right)^{\frac{v_1}{2} + k} \times \left(\frac{v_2}{v_2 + v_1 x} \right)^{\frac{v_1 + v_2}{2} + k} x^{\frac{v_1}{2} - 1 + k}$$

【例 4-15】非中心 F 分布随机数生成器示例。

```
>> r = ncfrnd(10,100,4,1,6)
```

```
r =
```

```
1.9742    0.9342    1.9818    0.8529    2.4364    0.8943
```

```
>> r1 = frnd(10,100,1,6)
```

```
r1 =
```

```
0.8195    0.3874    1.7207    0.6108    1.2783    0.5865
```

12. nctrnd 函数

功能：非中心 T 分布随机数生成器。其调用格式如下：

```
R = nctrnd(V,DELTA)
```

```
R = nctrnd(V,DELTA,v)
```

```
R = nctrnd(V,DELTA,m,n)
```

nctrnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x | v, \delta) = \frac{v^{v/2} e^{-v\delta^2/2(x^2+v)}}{\sqrt{x} \Gamma(v/2) 2^{(v-1)/2} (x^2 + v)^{(v+1)/2}}$$

【例 4-16】非中心 T 分布随机数生成器示例。

```
>> nctrnd(10,1,5)
```

```
ans =
```

```
0.4230    1.6596    1.1473    0.8878    0.3166
```

```
-0.2624    2.5349    1.6064    -1.4818    0.1486
```

```
-0.7716    0.2977    1.2458    1.3690    -0.2582
```

```
1.2165    1.9398    1.2791    1.8358    0.8410
```

```
1.2578    2.7610    0.7227    1.1217    0.1487
```

```
>> nctrnd(10,1,5,1)
```

```
ans =
```

```
-1.2265
```

```
2.8712
```

```
1.3638
```

```
3.2801
```

```
1.7474
```

13. ncx2rnd 函数

功能：非中心卡方分布随机数生成器。其调用格式如下：

$R = \text{ncx2rnd}(V, \text{DELTA})$

$R = \text{ncx2rnd}(V, \text{DELTA}, v)$

$R = \text{ncx2rnd}(V, \text{DELTA}, m, n)$

ncx2rnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|v, \delta) = \frac{1}{2} e^{-(x+\delta)/2} \left(\frac{x}{\delta}\right)^{v/4-1/2} J_{v/2-1}(\sqrt{\delta x})$$

【例 4-17】非中心卡方分布随机数生成器示例。

```
>> ncx2rnd(4,2,6)
ans =
    13.9976    3.3928    4.4475    14.4605    8.3327    6.0500
     4.1838    2.2663    7.4533    7.5647    5.0974    2.8543
     0.6944    1.8146    5.0382    3.1975    22.4927    2.6104
     7.2295    11.6120    2.7860    0.9413    8.2142    3.4675
     4.1543    0.3754    4.1345    9.0139    13.2860    9.0347
     4.3217    8.3028    7.5843    8.0480    1.2002    12.6059

>> ncx2rnd(4,2,6,3)
ans =
     3.2117     5.0030     7.5663
     6.8553    10.2931     4.3218
     0.6183     6.6642     6.3565
    10.3462     4.0421     1.8497
     4.1265     2.3548     7.4559
     4.3391     3.3335     5.4284
```

14. normrnd 函数

功能：正态分布随机数生成器。其调用格式如下：

$R = \text{normrnd}(\mu, \sigma)$

$R = \text{normrnd}(\mu, \sigma, v)$

$R = \text{normrnd}(\mu, \sigma, m, n)$

normrnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

【例 4-18】normrnd 函数应用示例。

```
>> n1 = normrnd(1:6, 1./(1:6))
n1 =
    -0.9488     2.5102     3.2872     4.0003     4.9858     5.5856
>> n2 = normrnd(0, 1, [1 5])
n2 =
     0.5812    -2.1924    -2.3193     0.0799    -0.9485
>> n3 = normrnd([1 2 3; 4 5 6], 0.1, 2, 3)
n3 =
```

1.0411	2.0858	3.0449
4.0677	4.9309	6.0101

【例 4-19】生成正态分布的随机数。

```
clc;
%设置正态分布的参数
mu0=log(1000);
sigma0=1;
%产生 len 个随机数
len=5;
y1=normrnd(mu0,sigma0,[1 len])
%产生 P 行 Q 列的矩阵
P=3;
Q=4;
y2=normrnd(mu0,sigma0,P,Q)
%显示正态分布的柱状图
M=1000;
y3=normrnd(mu0,sigma0,[1 M]);
figure(1);
t=0:0.1:max(y3);
hist(y3,t);
axis([0 max(y3) 0 50]);
xlabel('取值');
ylabel('计数值');
```

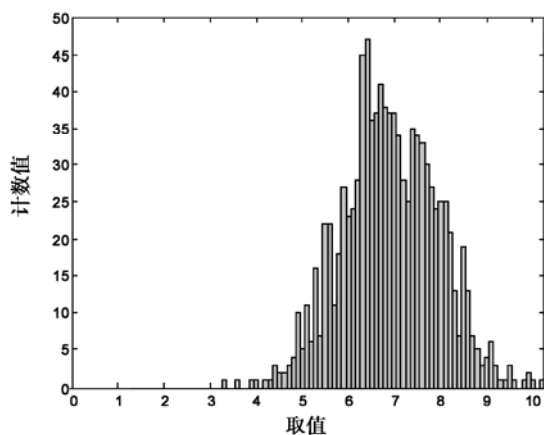


图 4-4 正态分布的柱状图

运行程序，输出如下，效果如图 4-4 所示。

```
y1 =
    6.4098    6.2500    6.8726    7.6707    6.6478
y2 =
    7.7830    6.4663    6.8112    6.8796
    6.6007    4.9262    6.7884    7.2607
    6.4438    7.9034    5.9953    9.7219
```

15. poissrnd 函数

功能：泊松分布随机数生成器。其调用格式如下：

$R = \text{poissrnd}(\text{lambda})$

$R = \text{poissrnd}(\text{lambda}, m)$

$R = \text{poissrnd}(\text{lambda}, m, n)$

poissrnd 函数相应分布的概率密度函数的数学公式如下:

$$f(x|\lambda) = \frac{\lambda^x}{x!} e^{-\lambda} I_{(0,1,\dots)}(x)$$

【例 4-20】 poissrnd 函数应用示例。

```
>> lambda = 2;
random_sample1 = poissrnd(lambda,1,10)
random_sample1 =
    3    3    1    2    3    2    0    0    1    2
>> random_sample2 = poissrnd(lambda,[1 10])
random_sample2 =
    4    2    2    0    5    2    3    2    2    1
>> random_sample3 = poissrnd(lambda(ones(1,10)))
random_sample3 =
    1    1    4    0    4    2    1    3    1    2
```

16. raylrnd 函数

功能: Rayleigh 分布的随机数生成器。其调用格式如下:

$R = \text{raylrnd}(B)$

$R = \text{raylrnd}(B, v)$

$R = \text{raylrnd}(B, m, n)$

raylrnd 函数相应分布的概率密度函数的数学公式如下:

$$f(x|b) = \frac{x}{b^2} e^{-\frac{x^2}{2b^2}}$$

【例 4-21】Rayleigh 分布的随机数生成器示例。

```
>> r = raylrnd(1:5)
r =
    1.7693    3.2055    3.0811    3.3193    1.4555
```

17. trnd 函数

功能: T 分布的随机数生成器。其调用格式如下:

$R = \text{trnd}(V)$

$R = \text{trnd}(v, m)$

$R = \text{trnd}(V, m, n)$

trnd 函数相应分布的概率密度函数的数学公式如下:

$$f(x|v) = \frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)} \frac{1}{\sqrt{v\pi}} \frac{1}{\left(1 + \frac{x^2}{v}\right)^{\frac{v+1}{2}}}$$

【例 4-22】T 分布的随机数生成器示例。

```
>> noisy = trnd(ones(1,6))
noisy =
    6.5206   -0.5895    6.8847    0.6083    0.7616    0.1267
>> numbers = trnd(1:6,[1 6])
numbers =
    0.2945   -0.2074    1.2258    0.8994   -2.3744   -1.6128
>> numbers = trnd(3,2,6)
numbers =
    1.2276   -3.1051    0.8344   -0.9943    0.3975    1.1765
   -0.1842   -0.7594   -0.0447   -1.2247   -1.0397   -0.4774
```

18. unidrnd 函数

功能：离散平均分布随机数生成器。其调用格式如下：

$R = \text{unidrnd}(N)$

$R = \text{unidrnd}(N,v)$

$R = \text{unidrnd}(N,m,n)$

unidrnd 函数相应分布的概率密度函数的数学公式如下：

$$f(x|N) = \frac{1}{N} I_{(1,2,\dots,N)}(x)$$

【例 4-23】离散平均分布随机数生成器示例。

```
>> numbers = unidrnd(10000,1,6)-1
numbers =
    3502    6620    4161    8419    8329    2564
>> numbers = unidrnd(10000,1,6)
numbers =
    6135    5823    5408    8700    2648    3181
```

19. unifrnd 函数

功能：连续平均分布随机生成器。其调用格式如下：

$R = \text{unifrnd}(A,B)$

$R = \text{unifrnd}(A,B,m,n,\dots)$

$R = \text{unifrnd}(A,B,[m,n,\dots])$

【例 4-24】生成连续均匀分布的随机数。

```
clc;
%设置连续均匀分布的参数
a=0;
b=30;
%产生 len 个随机数
len=5;
y1=unifrnd(a,b,[1 len])
%产生 P 行 Q 列的矩阵
P=3;
Q=4;
y2=unifrnd(a,b,P,Q)
%显示连续均匀分布的柱状图
M=1000;
```

```

y3=unifrnd(a,b,[1 M]);
figure(1);
t=min(y3):1:max(y3);
hist(y3,t);
axis([0 max(y3) 0 50]);
xlabel('取值');
ylabel('计数值');

```

运行程序，输出如下，效果如图 4-5 所示。

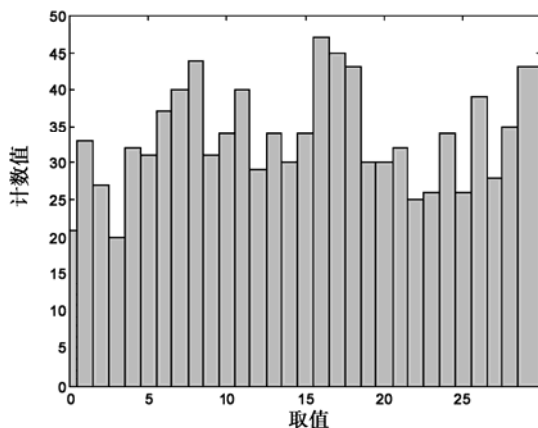


图 4-5 连续均匀分布的柱状图

```

y1 =
    12.5515    9.5448   28.0842   10.5440   17.3471
y2 =
    23.3204   11.7751    0.2377    7.7802
    14.1511    4.1326    1.4296    4.5320
    11.5254   25.2375   25.2972   20.1337

```

4.1.3 随机排序函数类型

在 MATLAB 中，randperm 函数可以用来生成 1 到 N 的一个随机序列，如：

```

>> randperm(8)
ans =
     1     4     8     6     5     3     7     2

```

同样地，randperm 函数的状态需要语句 “rand('state', m)” 来控制。利用这个函数可以对向量进行随机排序，如：

```

>> clear all;
a=[9 1 6 2 8 7 5 6 3];
rand('state',1); %固定随机数的状态
se=randperm(length(a));
ap=a(se)

```

运行程序，输出如下：

```

ap =
     3     7     6     2     1     5     8     9     6

```


ap 是 a 的一个随机排序。类似地，可以利用这个函数对矩阵的行或者列的顺序进行随机排列。下面的程序可以根据 se 来恢复 ap 到原来的向量 a。

```
>> [sp,ik]=sort(se);      %对 se 按从小到大顺序排列
ar=ap(ik)                 %用索引 ik 把 ap 恢复到原来的顺序
ar =
    9     1     6     2     8     7     5     6     3
```

可以发现向量 ar 与 a 是相等的。这里使用 sort 函数获得恢复至原来顺序的次序向量 ik。读者可以根据上面的方法对自己的数据进行排序处理。此外使用 randperm 函数还可以实现从 M 个数中选择 N 个数的目的，如：

```
>> clear all;
rand('state',111)        %设定随机函数状态
a=rand(1,9)               %产生 1×9 的一个随机数
ch1=randperm(9);          %产生 1:9 的一个随机排序
ch2=ch1(1:2);             %把 ch1 的前两个数赋值给 ch2
ch3=randperm(9);          %产生 1×9 的随机数
ch4=ch3(1:5);             %%把 ch3 的前五个数赋值给 ch4
b1=a(ch2)                 %从 a 中取出 ch2 代表位置处的 2 个数
b2=a(ch4)                 %从 a 中取出 ch4 代表位置处的 5 个数
```

运行程序，输出如下：

```
a =
    0.7140    0.4575    0.4630    0.6761    0.9799    0.3861    0.7791    0.1192    0.8715
b1 =
    0.4630    0.3861
b2 =
    0.6761    0.3861    0.8715    0.7791    0.4575
```

4.1.4 概率密度函数

给定 x 个随机变量，如果存在一个非负函数 $f(x)$ ，使得对任意实数 $a, b (a < b)$ 有 $P(a < X \leq b) = \int_a^b f(x)dx$ ，则称 $f(x)$ 为 x 的概率密度函数。 $f(x)$ 可以用来表示随机数的数字特征。概率密度函数具有下面两个性质：

$$(1) f(x) \geq 0$$

$$(2) \int_{-\infty}^{+\infty} f(x)dx = 1$$

利用概率密度函数的第 (2) 个性质可以计算出概率密度函数的未知参数。概率密度函数能够给出随机数或者数据取不同幅值大小的概率，在随机振动、随机疲劳实验等应用场合，常常利用它来检测采集数据的正态性质及了解幅值大小分布情况。

MATLAB 提供的计算概率密度函数有许多，下面展开介绍。

1. betapdf 函数

功能：beta 概率密度函数。其调用格式如下：

$Y = \text{betapdf}(X, A, B)$

betapdf 函数的概率密度数学公式为：

$$y=f(x|a,b)=\frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1}I_{(0,1)}(x)$$

【例 4-25】beta 概率密度函数示例。

```
>> a = [0.5 1; 2 4]
a =
    0.5000    1.0000
    2.0000    4.0000
>> y = betapdf(0.5,a,a)
y =
    0.6366    1.0000
    1.5000    2.1875
```

2. binopdf 函数

功能：二项式概率密度函数。其调用格式如下：

$Y = \text{binopdf}(X,N,P)$

binopdf 函数的概率密度数学公式为：

$$y=f(x|n,p)=\binom{n}{x}p^xq^{(n-x)}I_{(0,1,\dots,n)}(x)$$

【例 4-26】某人向空中抛硬币 100 次，落下为正面的概率为 0.5。这 100 次中正面向上的次数。其实现的 MATLAB 程序代码如下：

```
>> clear all;
p1=binopdf(45,100,0.5)    %计算 x=45 的概率
p2=binocdf(45,100,0.5)    %计算 x≤45 的概率即累积概率
x=1:100;
p=binopdf(x,100,0.5);
px=binopdf(x,100,0.5);
plot(x,p,'rp');           %绘制分布函数图像
figure;
plot(x,px,'+');           %绘制概率密度函数图像
```

运行程序，输出如下，效果如图 4-6 及图 4-7 所示。

```
p1 =
    0.0485
p2 =
    0.1841
```

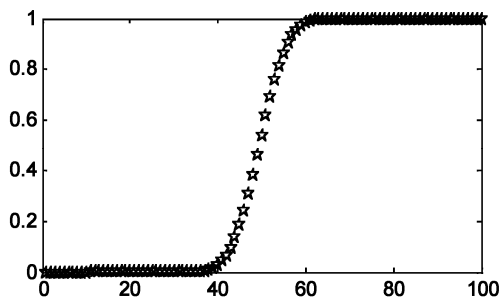


图 4-6 分布函数效果图

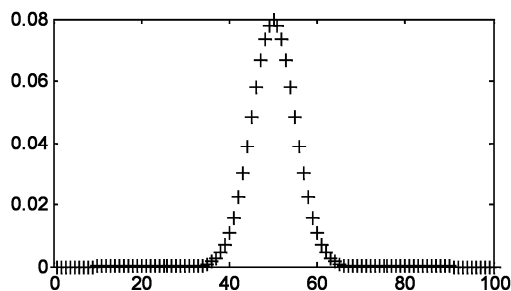


图 4-7 概率密度函数图像

【例 4-27】一个质检人员每天检查 200 块电路板，如果电路板 2% 有问题，那么这个质检人员在任何一天当中没有发现坏的电路板的概率是多少呢？

```
>> binopdf(0,200,0.02)
ans =
    0.0176
% 最多发现电路板有几块电路板有问题
>> defects=0:200;
y = binopdf(defects,200,.02);
[x,i]=max(y);
defects(i)
ans =
    4
```

3. chi2pdf 函数

功能：卡方分布的概率密度函数。其调用格式如下：

$Y = \text{chi2pdf}(X, V)$

【例 4-28】卡方分布的概率密度函数示例。

```
>> nu = 1:6;
x = nu;
y = chi2pdf(x,nu)
y =
    0.2420    0.1839    0.1542    0.1353    0.1220    0.1120
```

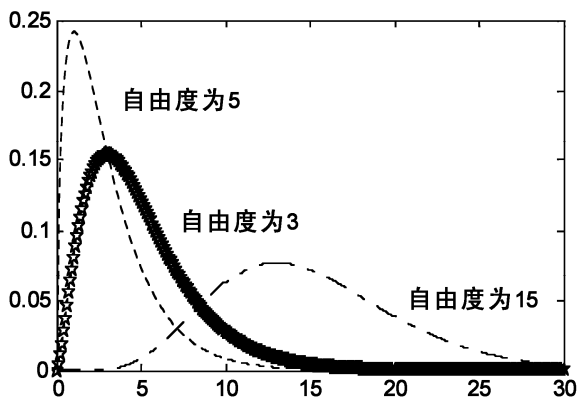
【例 4-29】分别绘制自由度 $n=1, 5, 15$ 的 χ^2 分布概率密度函数曲线，并求出自由度为 15 的 χ^2 分布的均值与方差。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x=0:0.1:30;          %给出 x 的取值
y1=chi2pdf(x,3);      %计算出对应于 x 的自由度 1 的概率密度函数数值
plot(x,y1,'r:');
hold on;
y2=chi2pdf(x,5);
plot(x,y2,'kp');
y3=chi2pdf(x,15);
plot(x,y3,'b-.');
gtext('自由度为 3');
gtext('自由度为 5');
gtext('自由度为 15');
axis([0 30 0 0.25])   %指定显示的图形区域
[m,v]=chi2stat(15)
```

运行程序，输出如下，效果如图 4-8 所示。

```
m =    15
v =    30
```

图 4-8 χ^2 分布密度曲线

4. exppdf 函数

功能：指数分布的概率密度函数。其调用格式如下：

$Y = \text{exppdf}(X, \mu)$

exppdf 函数的概率密度数学公式为：

$$y = f(x|\mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

【例 4-30】exppdf 函数用法示例。

```
>> y = exppdf(5,1:5)
y =
    0.0067    0.0410    0.0630    0.0716    0.0736
>> y = exppdf(1:5,1:5)
y =
    0.3679    0.1839    0.1226    0.0920    0.0736
```

5. fpdf 函数

功能：F 分布概率密度函数。其调用格式如下：

$Y = \text{fpdf}(X, V1, V2)$

【例 4-31】F 分布概率密度函数示例。

```
>> y = fpdf(1:6,2,2)
y =
    0.2500    0.1111    0.0625    0.0400    0.0278    0.0204
>> z = fpdf(3,5:10,5:10)
z =
    0.0689    0.0659    0.0620    0.0577    0.0532    0.0487
```

6. gampdf 函数

功能：gamma 概率密度函数。其调用格式如下：

$Y = \text{gampdf}(X, A, B)$

【例 4-32】gamma 概率密度函数示例。

```
>> mu = 1:5;
y = gampdf(1,1,mu)
```

```
y =
    0.3679    0.3033    0.2388    0.1947    0.1637
>> y1 = exppdf(1,mu)
y1 =
    0.3679    0.3033    0.2388    0.1947    0.1637
```

7. geopdf 函数

功能：几何分布的概率密度函数。其调用格式如下：

$Y = \text{geopdf}(X,P)$

【例 4-33】如果抛一次硬币，正面朝上是成功，否则失败，那么在第一次成功以前有三次失败的概率是多少？

```
>> p = geopdf(3,0.5)
p =
    0.0625
```

8. hygepdf 函数

功能：超几何分布概率密度函数。其调用格式如下：

$Y = \text{hygepdf}(X,M,K,N)$

【例 4-34】如果 100 张软盘，其中 20 张是坏盘，那么抽 10 张出来，坏盘的张数是 0~5 张的概率分别是多少呢？

```
>> p = hygepdf(0:5,100,20,10)
p =
    0.0951    0.2679    0.3182    0.2092    0.0841    0.0215
```

9. lognpdf 函数

功能：对数正态分布概率密度函数。其调用格式如下：

$Y = \text{lognpdf}(X,\mu,\sigma)$

【例 4-35】对数正态分布概率密度函数示例。

```
>> x = (0:0.02:10);
y = lognpdf(x,0,1);
plot(x,y); grid;
xlabel('x'); ylabel('p')
title('对数正态分布概率密度');
```

运行程序，效果如图 4-9 所示。

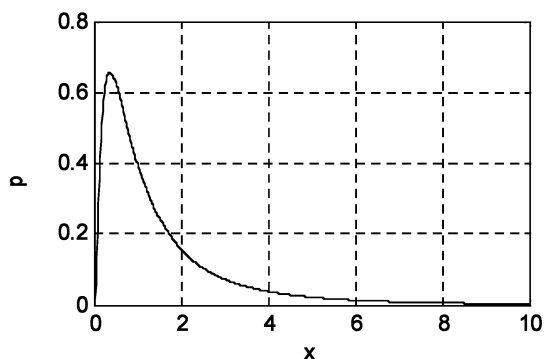


图 4-9 对数正态分布概率密度函数

10. nbinpdf 函数

功能：负二项分布概率密度函数。其调用格式如下：

$Y = \text{nbpdf}(X, R, P)$

【例 4-36】负二项分布概率密度函数示例。

```
>> clear all;
x = (0:10);
y = nbpdf(x,3,0.5);
plot(x,y,'rp')
set(gca,'Xlim',[-0.5,10.5])
title('负二项分布概率密度');
```

运行程序，效果如图 4-10 所示。

11. ncfpdf 函数

功能：非中心 F 分布概率密度函数。其调用格式如下：

$Y = \text{ncfpdf}(X, \text{NU1}, \text{NU2}, \text{DELTA})$

【例 4-37】非中心 F 分布概率密度函数示例。

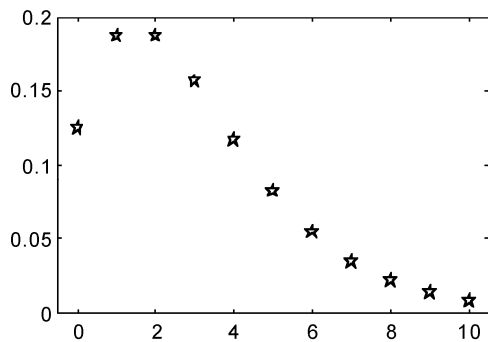


图 4-10 负二项分布概率密度函数

```
>> clear all;
x = (0.01:0.1:10.01)';
p1 = ncfpdf(x,5,20,10);
p = fpdf(x,5,20);
plot(x,p,'-',x,p1,'-')
title('非中心 F 分布概率密度');
```

运行程序，效果如图 4-11 所示。

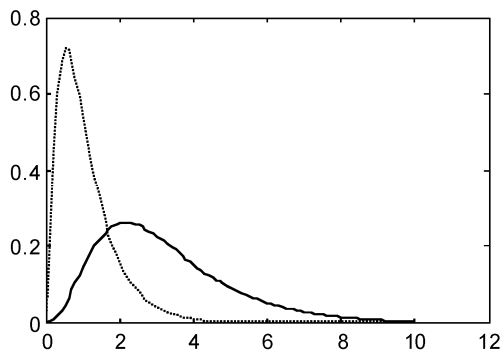


图 4-11 非中心 F 分布概率密度函数

12. nctpdf 函数

功能：非中心 T 分布概率密度函数。其调用格式如下：

$Y = \text{nctpdf}(X, V, \text{DELTA})$

【例 4-38】非中心 T 分布概率密度函数示例。

```
>> clear all;
x = (-5:0.1:5)';
nct = nctpdf(x,10,1);
t = tpdf(x,10);
plot(x,nct,'r','LineWidth',2)
hold on
plot(x,t,'g-','LineWidth',2)
legend('nct','t')
title('非中心 T 分布概率密度');
```

运行程序，效果如图 4-12 所示。

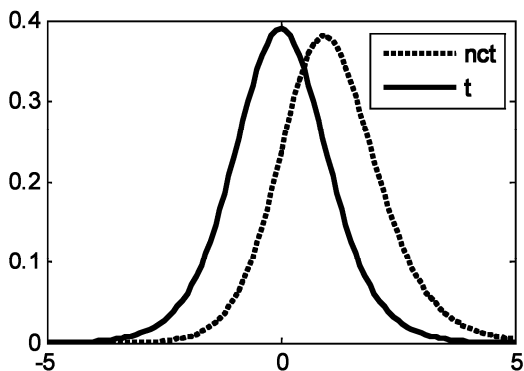


图 4-12 非中心 T 分布概率密度函数

13. ncx2pdf 函数

功能：非中心卡方分布概率密度函数。其调用格式如下：

$Y = \text{ncx2pdf}(X, V, \text{DELTA})$

【例 4-39】非中心卡方分布概率密度函数示例。

```
>> clear all;
x = (0:0.1:10)';
ncx2 = ncx2pdf(x,4,2);
chi2 = chi2pdf(x,4);
plot(x,ncx2,'r','LineWidth',2)
hold on
plot(x,chi2,'g-','LineWidth',2)
legend('ncx2','chi2')
title('非中心卡方分布概率密度');
```

运行程序，效果如图 4-13 所示。

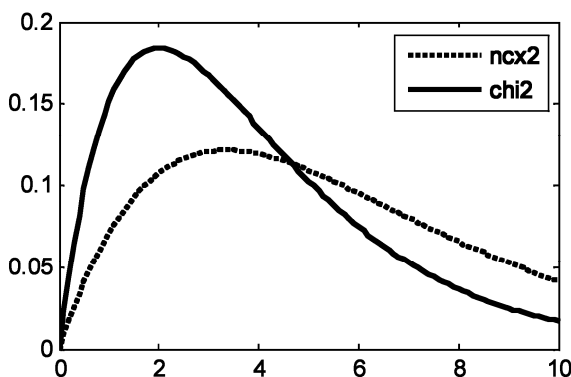


图 4-13 非中心卡方分布概率密度函数

14. normpdf 函数

功能：正态分布概率密度函数。其调用格式如下：

$Y = \text{normpdf}(X, \mu, \sigma)$

【例 4-40】正态分布参数对密度曲线的影响。

%设定正态分布的分布参数 μ 和 σ 及绘图区域

```
clear all;
```

```
mu1=2.5;
```

```
mu2=3;
```

```
sigma1=0.5;
```

```
sigma2=0.6;
```

```
x=(mu2-4*sigma2):0.01:(mu2+4*sigma2);
```

%考察 μ 的影响

```
y1=normpdf(x,mu1,sigma1);
```

```
y2=normpdf(x,mu2,sigma1);
```

%考察 σ 的影响

```
y3=normpdf(x,mu1,sigma1);
```

```
y4=normpdf(x,mu1,sigma2);
```

%考察结果的可视化

```
subplot(1,2,1)
```

```
plot(x,y1,'r:',x,y2,'-b')
```

```
xlabel('\fontsize{12} $\mu 1 < \mu 2, \sigma 1 = \sigma 2$ ')

```

```
legend('  $\mu 1$  ', '  $\mu 2$  ')

```

```
subplot(1,2,2)
```

```
plot(x,y3,'r:',x,y4,'-b')
```

```
xlabel('\fontsize{12}  $\mu 1 = \mu 2, \sigma 1 < \sigma 2$ ')

```

```
legend('  $\sigma 1$  ', '  $\sigma 2$  ')

```

运行程序，效果如图 4-14 所示。

15. poisspdf 函数

功能：泊松分布概率密度函数。其调用格式如下：

$Y = \text{poisspdf}(X, \lambda)$

【例 4-41】分别绘制出 $\lambda=1,2,5,10$ 时泊松分布的概率密度函数曲线。

```
>> clear all;
```

```
x=[0:15]';
```



```
y=[];
lam1=[1,2,5,10];
for i=1:length(lam1)
    y=[y,poisspdf(x,lam1(i))];
end
plot(x,y)
title('泊松分布概率密度');
```

运行程序，效果如图 4-15 所示。

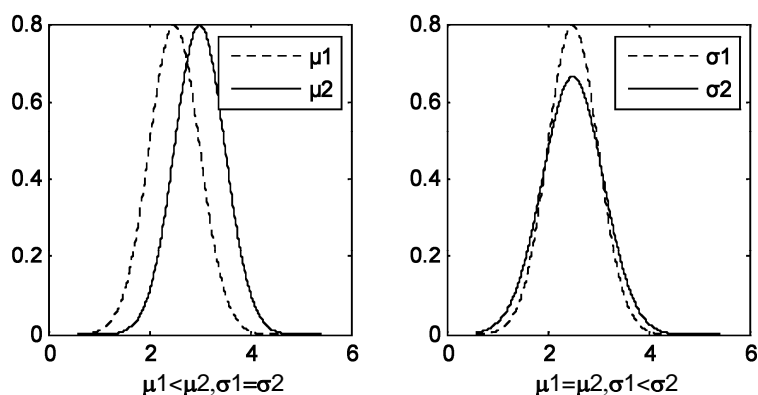


图 4-14 正态分布参数对密度曲线的影响

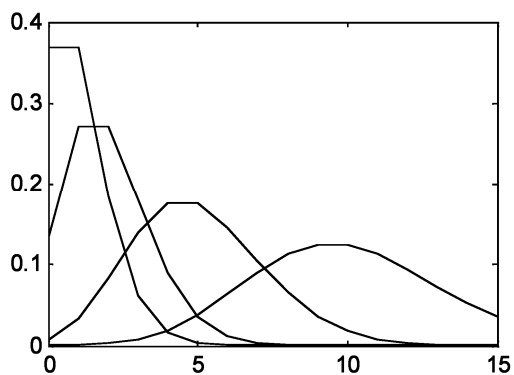


图 4-15 泊松分布的概率密度函数

16. raylpdf 函数

功能：Rayleigh 分布的概率密度函数。其调用格式如下：

$Y = \text{raylpdf}(X,B)$

【例 4-42】Rayleigh 分布的概率密度函数示例。

```
>> clear all;
x = 0:0.1:3;
p = raylpdf(x,1);
plot(x,p)
title('Rayleigh 分布的概率密度')
```

运行程序，效果如图 4-16 所示。

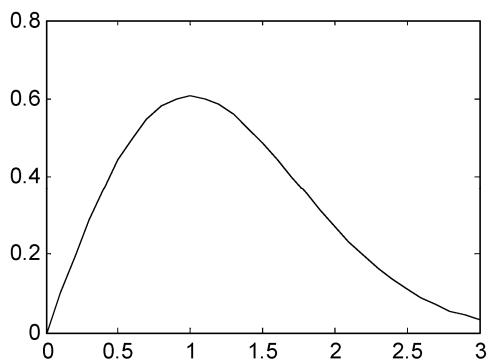


图 4-16 Rayleigh 分布的概率密度函数

17. tpdf 函数

功能：T 分布的概率密度函数。其调用格式如下：

$Y = \text{tpdf}(X,V)$

【例 4-43】T 分布的概率密度函数示例。

```
>> tpdf(0,1:6)
ans =
    0.3183    0.3536    0.3676    0.3750    0.3796    0.3827
>> difference = tpdf(-2.5:2.5,30)-normpdf(-2.5:2.5)
difference =
    0.0035   -0.0006   -0.0042   -0.0042   -0.0006    0.0035
```

18. unidpdf 函数

功能：离散平均分布概率密度函数。其调用格式如下：

$Y = \text{unidpdf}(X,N)$

【例 4-44】离散平均分布概率密度函数示例。

```
>> y = unidpdf(1:6,10)
y =
    0.1000    0.1000    0.1000    0.1000    0.1000    0.1000
>> likelihood = unidpdf(5,4:9)
likelihood =
     0    0.2000    0.1667    0.1429    0.1250    0.1111
```

19. unifpdf 函数

功能：连续均匀分布概率密度函数。其调用格式如下：

$Y = \text{unifpdf}(X,A,B)$

【例 4-45】连续均匀分布概率密度函数示例。

```
>> x = 0.1:0.1:0.6;
y = unifpdf(x)
y =
     1     1     1     1     1     1
>> y = unifpdf(-1,0,1)
y =
     0
```

4.1.5 累积概率值

累积概率值可以用来计算产品或者试样在规定条件（或者规定参数）下符合要求和不符合要求的概率。因此借助这个参数可以用来检查产品的生产情况、比较方案优劣、确定项目可行性和风险程度，MATLAB 提供的计算累积概率值分别作如下介绍。

1. betacdf 函数

功能：beta 累积分布函数。其调用格式如下：

$p = \text{betacdf}(X,A,B)$

betacdf 函数的概率数学公式为

$$p = F(x|a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1} (1-t)^{b-1} dt$$

【例 4-46】beta 累积分布函数示例。

```
>> x = 0.1:0.2:0.9;
a = 2;
b = 2;
p = betacdf(x,a,b)
p =
    0.0280    0.2160    0.5000    0.7840    0.9720
>> a = [1 2 3];
p = betacdf(0.5,a,a)
p =
    0.5000    0.5000    0.5000
```

2. binocdf 函数

功能：二项式累积分布函数。其调用格式如下：

$Y = \text{binocdf}(X,N,P)$

binocdf 函数的概率数学公式为

$$y = F(x|n,p) = \sum_{i=0}^x \binom{n}{i} p^i q^{(n-i)} I_{(0,1,\dots,n)}(i)$$

【例 4-47】如果一个棒球队在一个赛季要打 158 场球赛，如果胜负的概率各半，则他们赢球的概率为

```
>> 1 - binocdf(100,162,0.5)
ans =
    0.0010
```

3. chi2cdf 函数

功能：卡方分布的累积分布函数。其调用格式如下：

$P = \text{chi2cdf}(X,V)$

chi2cdf 函数的概率数学公式为

$$p = F(x|v) = \int_0^x \frac{t^{(v-2)/2} e^{-t/2}}{2^{v/2} \Gamma(v/2)} dt$$

【例 4-48】卡方分布的累积分布函数示例。

```
>> probability = chi2cdf(5,1:5)
```

```
probability =
    0.9747    0.9179    0.8282    0.7127    0.5841
>> probability = chi2cdf(1:5,1:5)
probability =
    0.6827    0.6321    0.6084    0.5940    0.5841
```

4. expcdf 函数

功能：指数分布的累积分布函数。其调用格式如下：

$P = \text{expcdf}(X, \mu)$

$[P, PLO, PUP] = \text{expcdf}(X, \mu, \text{pcov}, \alpha)$

expcdf 函数的概率数学公式为

$$p = F(x | \mu) = \int_0^x \frac{1}{\mu} e^{-\frac{t}{\mu}} dt = 1 - e^{-\frac{x}{\mu}}$$

【例 4-49】指数分布的累积分布函数示例。

```
>> mu = 10:10:60;
p = expcdf(log(2)*mu,mu)
p =
    0.5000    0.5000    0.5000    0.5000    0.5000    0.5000
>> mu = 1:6;
x = mu;
p = expcdf(x,mu)
p =
    0.6321    0.6321    0.6321    0.6321    0.6321    0.6321
```

5. fcdf 函数

功能：F 分布累积分布函数。其调用格式如下：

$P = \text{fcdf}(X, V1, V2)$

fcdf 函数的概率数学公式为

$$p = F(x | v_1, v_2) = \int_0^x \frac{\Gamma\left(\frac{v_1 + v_2}{2}\right)}{\Gamma\left(\frac{v_1}{2}\right)\Gamma\left(\frac{v_2}{2}\right)} \left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}} \frac{t^{\frac{v_1}{2}-2}}{\left[1 + \left(\frac{v_1}{v_2}\right)t\right]^{\frac{v_1 + v_2}{2}}} dt$$

【例 4-50】F 分布累积分布函数示例。

```
>> nu1 = 1:5;
nu2 = 6:10;
x = 2:6;
F1 = fcdf(x,nu1,nu2)
F1 =
    0.7930    0.8854    0.9481    0.9788    0.9919
>> F2 = 1 - fcdf(1./x,nu2,nu1)
F2 =
    0.7930    0.8854    0.9481    0.9788    0.9919
```

6. gamcdf 函数

功能：gamma 累积分布函数。其调用格式如下：

gamcdf(X,A,B)

[P,PLO,PUP] = gamcdf(X,A,B,pcov,alpha)

gamcdf 函数的概率数学公式为

$$p = F(x|a,b) = \frac{1}{b^a \Gamma(a)} \int_0^x t^{a-1} e^{-\frac{t}{b}} dt$$

【例 4-51】 gamma 累积分布函数示例。

```
>> a = 1:6;
b = 5:10;
prob = gamcdf(a.*b,a,b)
prob =
    0.6321    0.5940    0.5768    0.5665    0.5595    0.5543
```

7. geocdf 函数

功能：几何分布的累积分布函数。其调用格式如下：

Y = geocdf(X,P)

geocdf 函数的概率数学公式为

$$y = F(x|p) = \sum_{i=0}^{\text{floor}(x)} p q^i, \quad q = 1 - p$$

【例 4-52】 几何分布的累积分布函数示例。

```
>> p = geocdf(3,0.5)
p =
    0.9375
```

8. hygecdf 函数

功能：超几何分布累积分布函数。其调用格式如下：

hygecdf(X,M,K,N)

hygecdf 函数的概率数学公式为

$$p = F(x|M,K,N) = \sum_{i=0}^x \frac{\binom{K}{i} \binom{M-K}{N-i}}{\binom{M}{N}}$$

【例 4-53】 如果 200 张软盘，其中 50 张是坏盘，那么抽 20 张，坏盘的张数是 0~5 张的概率是多少？

```
>> p=hygecdf(5,200,50,20)
p =
    0.6200
```

9. logncdf 函数

功能：对数正态分布累积分布函数。其调用格式如下：

P = logncdf(X,mu,sigma)

[P,PLO,PUP] = logncdf(X,mu,sigma,pcov,alpha)

logncdf 函数的概率数学公式为

$$p = F(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^x e^{-\frac{(\ln(t) - \mu)^2}{2\sigma^2}} dt$$

【例 4-54】对数正态分布累积分布函数示例。

```
>> clear all;
x = (0:0.2:10);
y = logncdf(x,0,1);
plot(x,y,'r');
grid on;
xlabel('x'); ylabel('p');
title('对数正态分布累积分布')
```

运行程序，效果如图 4-17 所示。

10. nbincdf 函数

功能：负二项分布累积分布函数。其调用格式如下：

$Y = \text{nbincdf}(X, R, P)$

nbincdf 函数的概率数学公式为

$$y = F(x|r, p) = \sum_{i=0}^x \binom{r+i-1}{i} p^r q^i I_{(0,1,\dots)}(i)$$

【例 4-55】负二项分布累积分布函数示例。

```
>> clear all;
x = (0:15);
p = nbincdf(x,3,0.5);
stairs(x,p)
xlabel('x'); ylabel('p');
title('负二项分布累积分布')
```

运行程序，效果如图 4-18 所示。

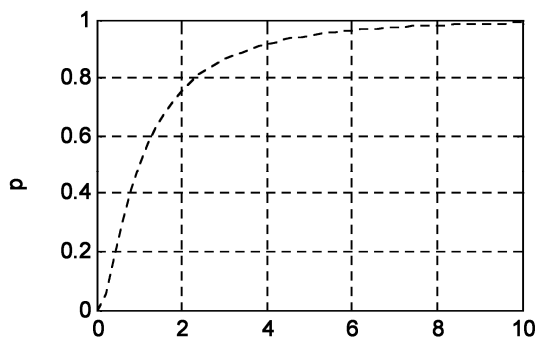


图 4-17 对数正态分布累积分布函数

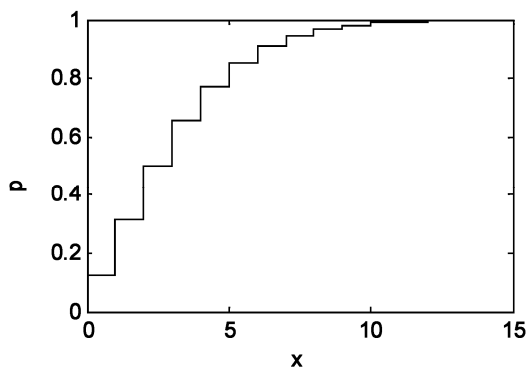


图 4-18 负二项分布累积分布函数

11. nfcdf 函数

功能：非中心 F 分布累积分布函数。其调用格式如下

$P = \text{nfcdf}(X, \text{NU1}, \text{NU2}, \text{DELTA})$

nfcdf 函数的概率数学公式为：

$$f(x|v_1, v_2, \delta) = \sum_{j=0}^{\infty} \left[\frac{\left(\frac{1}{2}\delta\right)^j}{j!} e^{-\frac{\delta}{2}} \right] I \left[\frac{v_1 \cdot x}{v_2 + v_1 \cdot x} \middle| \frac{v_1}{2} + j \cdot \frac{v_2}{2} \right]$$

【例 4-56】非中心 F 分布累积分布函数示例。

```
>> clear all;
x = (0.01:0.1:10.01)';
p1 = ncfcdf(x,5,20,10);
p = fcdf(x,5,20);
plot(x,p,'r',x,p1,'g-')
title('非中心 F 分布累积分布')
```

运行程序，效果如图 4-19 所示。

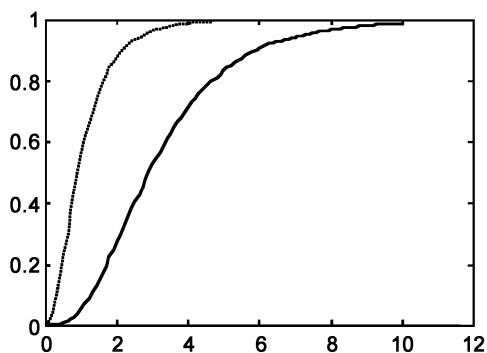


图 4-19 非中心 F 分布累积分布函数

12. nctcdf 函数

功能：非中心 T 分布累积分布函数。其调用格式如下：

$P = \text{nctcdf}(X, NU, DELTA)$

【例 4-57】与 T 分布做比较。

```
>> clear all;
x = (-5:0.1:5)';
p1 = nctcdf(x,10,1);
p = tcdf(x,10);
plot(x,p,'r',x,p1,'-')
legend('非中心 T 分布','中心 T 分布')
```

运行程序，效果如图 4-20 所示。

13. ncx2cdf 函数

功能：非中心卡方分布累积分布函数。其调用格式如下：

$P = \text{ncx2cdf}(X, V, DELTA)$

ncx2cdf 函数的概率数学公式为

$$F(x|v, \delta) = \sum_{j=0}^{\infty} \left[\frac{\left(\frac{1}{2}\delta\right)^j}{j!} e^{-\frac{\delta}{2}} \right] \Pr[x_{v+2j}^2 \leq x]$$

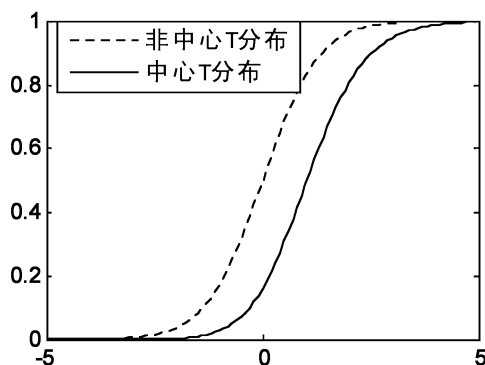


图 4-20 非中心 T 分布累积分布函数

【例 4-58】非中心卡方分布累积分布函数示例。

```
>> clear all;
x = (0:0.1:10)';
ncx2 = ncx2cdf(x,4,2);
chi2 = chi2cdf(x,4);
plot(x,ncx2,'b-', 'LineWidth',2)
hold on
plot(x,chi2,'g--','LineWidth',2)
legend('ncx2','chi2','Location','NW')
```

运行程序，效果如图 4-21 所示。

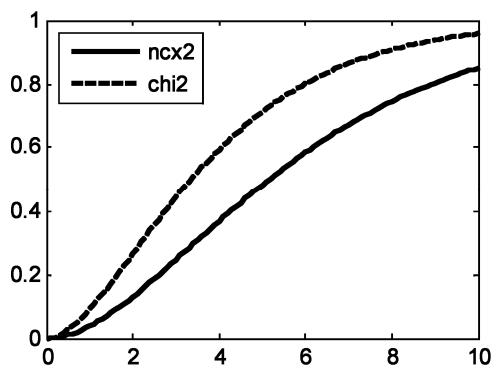


图 4-21 非中心卡方分布累积分布函数

14. normcdf 函数

功能：正态分布累积分布函数。其调用格式如下：

$P = \text{normcdf}(X, \mu, \sigma)$

$[P, PLO, PUP] = \text{normcdf}(X, \mu, \sigma, \text{pcov}, \alpha)$

normcdf 函数的概率数学公式为

$$p = F(x | \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

【例 4-59】标准正态分布在区间[-1 1]失败的概率是多少？

```
>> p = normcdf([-1 1]);
p(2)-p(1)
ans =
    0.6827
```

15. poisscdf 函数

功能：泊松分布累积分布函数。其调用格式如下：

$P = \text{poisscdf}(X, \text{lambda})$

poisscdf 函数的概率数学公式为

$$p = F(x | \lambda) = e^{-\lambda} \sum_{i=0}^{\text{floor}(x)} \frac{\lambda^i}{i!}$$

【例 4-60】泊松分布累积分布函数示例。

```
>> probability = 1-poisscdf(4,2)
probability =
    0.0527
>> probability = poisscdf(4,4)
probability =
    0.6288
```

16. raylcdf 函数

功能：Rayleigh 分布的累积分布函数。其调用格式如下：

$P = \text{raylcdf}(X, B)$

raylcdf 函数的概率数学公式为

$$y = F(x | b) = \int_0^x \frac{t}{b^2} e^{\left(\frac{-t^2}{2b^2}\right)} dt$$

【例 4-61】Rayleigh 分布的累积分布函数示例。

```
>> clear all;
x = 0:0.1:3;
p = raylcdf(x,1);
plot(x,p)
title('Rayleigh 分布的累积分布')
```

运行程序，效果如图 4-22 所示。

17. tcdf 函数

功能：T 分布。其调用格式如下：

$P = \text{tcdf}(X, V)$

tcdf 函数的概率数学公式为

$$p = F(x|v) = \int_{-\infty}^x \frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)} \frac{1}{\sqrt{v\pi}} \frac{1}{\left(1 + \frac{t^2}{v}\right)^{\frac{v+1}{2}}} dt$$

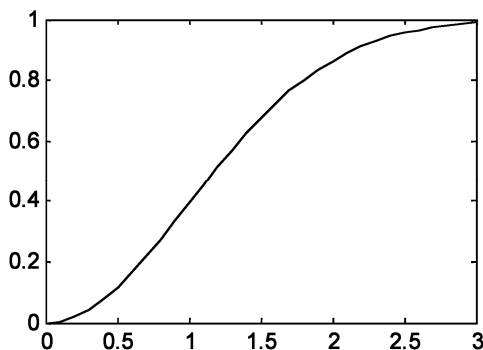


图 4-22 Rayleigh 分布的累积分布函数

【例 4-62】T 分布示例。

```
>> mu = 1;
sigma = 2;
n = 100;
x = normrnd(mu,sigma,n,1);
xbar = mean(x);
s = std(x);
t = (xbar-mu)/(s/sqrt(n))
p = 1-tcdf(t,n-1)
t =
    1.0589
p =
    0.1461
```

18. unidcdf 函数

功能：离散平均分布累积分布函数。其调用格式如下：

$P = \text{unidcdf}(X,N)$

unidcdf 函数的概率数学公式为

$$p = F(x|N) = \frac{\text{floor}(x)}{N} (1, 2, \dots, N)(x)$$

【例 4-63】从 1~50 的数中选出一个数，这个数小于等于 30 的概率是多少？

```
>> probability = unidcdf(30,50)
probability =
    0.6000
```

19. unifcdf 函数

功能：连续均匀分布累积分布函数。其调用格式如下：

$$p = F(x|a,b) = \frac{x-a}{b-a} I_{[a,b]}(x)$$

【例 4-64】连续均匀分布累积分布函数示例。

```
>> probability = unifcdf(0.75)
probability =
    0.7500
>> probability = unifcdf(0.75,-1,1)
probability =
    0.8750
```

4.1.6 逆累积分布函数

在已知概率值求解该概率的分布点时，就要用到逆累积分布函数。利用这个函数，可以根据概率值计算相关参数的取值，从而对测试的性能等指标进行有效的估计。MATLAB 提供了相关计算逆累积分布函数。分别介绍如下：

1. betainv 函数

功能：beta 临界值。其调用格式如下：

$X = \text{betainv}(P,A,B)$

betainv 函数的逆累积分布公式如下：

$$x = F^{-1}(p | a, b) = \{x : F(x | a, b) = p\}$$

其中

$$p = F(x | a, b) = \frac{1}{B(a, b)} \int_0^x t^{a-1} (1-t)^{b-1} dt$$

【例 4-65】beta 临界值示例。

```
>> p = [0.01 0.5 0.99];
x = betainv(p,10,5)
x =
    0.3726    0.6742    0.8981
```

2. binoinv 函数

功能：二项式临界值。其调用格式如下：

$X = \text{binoinv}(Y,N,P)$

【例 4-66】在一个胜负各半的比赛中，一个队在一个赛季中合理地胜的场次是多少（假设总数为 162 场）？

```
>> binoinv([0.05 0.95],162,0.5)
ans =
    71    91
```

3. chi2inv 函数

功能：卡方分布的临界值。其调用格式如下：

$X = \text{chi2inv}(P,V)$

chi2inv 函数的逆累积分布公式如下：

$$x = F^{-1}(p | v) = \{x : F(x | v) = p\}$$

其中



$$p = F(x|v) = \int_0^x \frac{t^{(v-2)/2} e^{-t/2} dt}{2^{v/2} \Gamma(v/2)}$$

【例 4-67】卡方分布的临界值示例。

```
>> x = chi2inv(0.95,10)
x =
    18.3070
```

4. expinv 函数

功能：指数分布的临界值。其调用格式如下：

$X = \text{expinv}(P, \mu)$

$[X, XLO, XUP] = \text{expinv}(X, \mu, \text{pcov}, \alpha)$

expinv 函数的逆累积分布公式如下：

$$x = F(p|\mu) = -\mu \ln(1-p)$$

【例 4-68】指数分布的临界值示例。

```
>> expinv(0.50,700)
ans =
    485.2030
```

5. finv 函数

功能：F 分布临界值。其调用格式如下：

$X = \text{finv}(P, V1, V2)$

finv 函数的逆累积分布公式如下：

$$x = F^{-1}(p|v_1, v_2) = \{x : F(x|v_1, v_2) = p\}$$

其中

$$p = F(x|v_1, v_2) = \int_0^x \frac{\Gamma\left[\left(\frac{v_1 + v_2}{2}\right)\right]}{\Gamma\left(\frac{v_1}{2}\right)\Gamma\left(\frac{v_2}{2}\right)} \left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}} \frac{t^{\frac{v_1-2}{2}}}{\left[1 + \left(\frac{v_1}{v_2}\right)t\right]^{\frac{v_1+v_2}{2}}} dt$$

【例 4-69】F 分布临界值示例。

```
>> x = finv(0.95,5,10)
x =
    3.3258
```

6. gaminv 函数

功能：gamma 临界值。其调用格式如下：

$X = \text{gaminv}(P, A, B)$

$[X, XLO, XUP] = \text{gamcdf}(P, A, B, \text{pcov}, \alpha)$

gaminv 函数的逆累积分布公式如下：

$$x = F^{-1}(p|a, b) = \{x : F(x|a, b) = p\}$$

其中



$$p = F(x|a,b) = \frac{1}{b^a \Gamma(a)} \int_0^x t^{a-1} e^{-\frac{t}{b}} dt$$

【例 4-70】演示了 gamma 分布和它的逆分布的关系。

```
>> a = 1:5;
b = 6:10;
x = gaminv(gamcdf(1:5,a,b),a,b)
x =
    1.0000    2.0000    3.0000    4.0000    5.0000
```

7. geoinv 函数

功能：几何分布的临界值。其调用格式如下：

$X = \text{geoinv}(Y,P)$

【例 4-71】连续猜对 10 枚硬币的正反面的概率小于 0.001。

```
>> psychic = geoinv(0.999,0.5)
psychic =
     9
```

8. hygeinv 函数

功能：超几何分布临界值。其调用格式如下：

$\text{hygeinv}(P,M,K,N)$

【例 4-72】超几何分布临界值示例。

```
>> x = hygeinv(0.99,1000,10,50)
x =
     3
>> x = hygeinv(0.50,1000,10,50)
x =
     0
```

9. icdf 函数

功能：计算逆累积分布函数。其调用格式如下：

$Y = \text{icdf}(\text{name},X,A)$

$Y = \text{icdf}(\text{name},X,A,B)$

$Y = \text{icdf}(\text{name},X,A,B,C)$

【例 4-73】计算逆累积分布函数示例。

```
>> x1 = icdf('Normal',0.1:0.2:0.9,0,1)
x1 =
    -1.2816    -0.5244         0     0.5244     1.2816
>> x2 = icdf('Poisson',0.1:0.2:0.9,0:4)
x2 =
    NaN         0         2         4         7
```

10. logninv 函数

功能：对数正态分布临界值。其调用格式如下：

$X = \text{logninv}(P,\mu,\sigma)$

$[X,XLO,XUP] = \text{logninv}(P,\mu,\sigma,\text{pcov},\alpha)$

logninv 函数的逆累积分布公式如下：

$$x = F^{-1}(p | \mu, \sigma) = \{x : F(x | \mu, \sigma) = p\}$$

其中

$$p = F(x | \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

【例 4-74】对数正态分布临界值示例。

```
>> clear all;
p = (0.005:0.01:0.995);
crit = logninv(p,1,0.5);
plot(p,crit)
xlabel('概率'); ylabel('临界值');
grid on;
title('对数正态分布临界值');
```

运行程序，效果如图 4-23 所示。

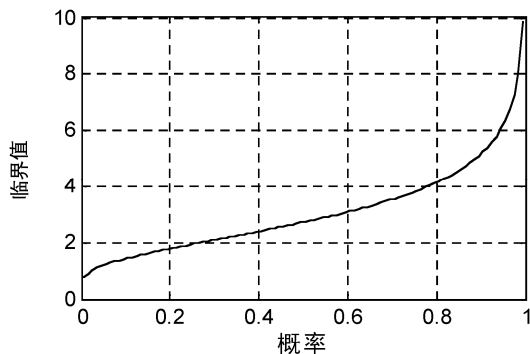


图 4-23 对数正态分布临界值效果

11. nbininv 函数

功能：负二项分布临界值。其调用格式如下：

$X = \text{nbininv}(Y,R,P)$

【例 4-75】要有 98% 的概率见到硬币的正面，你需要抛几次硬币？

```
>> flips = nbininv(0.98,10,0.5) + 10
flips =
    31
```

12. ncx2inv 函数

功能：非中心卡方分布临界值。其调用格式如下：

$X = \text{ncx2inv}(P,V,DELTA)$

【例 4-76】非中心卡方分布临界值示例。

```
>> ncx2inv([0.01 0.05 0.1],4,2)
ans =
    0.4858    1.1498    1.7066
```

13. ncfinv 函数

功能：非中心 F 分布临界值。其调用格式如下：

$X = \text{ncfinv}(P,NU1,NU2,DELTA)$

【例 4-77】非中心 F 分布临界值示例。

```
>> critical = finv(0.95,5,20)
critical =
    2.7109
>> prob = 1 - ncfcdf(critical,5,20,2)
prob =
    0.1297
>> ncfinv(0.5,5,20,2)
ans =
    1.2786
```

14. nctinv 函数

功能：非中心 T 的分布临界值。其调用格式如下：

$X = \text{nctinv}(P, NU, DELTA)$

【例 4-78】非中心 T 的分布临界值示例。

```
>> x = nctinv([0.1 0.2],10,1)
x =
   -0.2914    0.1618
```

15. norminv 函数

功能：正态分布临界值。其调用格式如下：

$X = \text{norminv}(P, \mu, \sigma)$

$[X, XLO, XUP] = \text{norminv}(P, \mu, \sigma, \text{pcov}, \alpha)$

norminv 函数的逆累积分布公式如下：

$$x = F^{-1}(p | \mu, \sigma) = \{x : F(x | \mu, \sigma) = p\}$$

其中

$$p = F(x | \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

【例 4-79】正态分布临界值示例。

```
>> x = norminv([0.025 0.975],0,1)
x =
   -1.9600    1.9600
>> xl = norminv([0.01 0.96],0,1)
xl =
   -2.3263    1.7507
```

16. poissinv 函数

功能：泊松分布临界值。其调用格式如下：

$X = \text{poissinv}(P, \lambda)$

【例 4-80】设 $X \sim N(2, \sigma^2)$ 。

(1) 当 $\sigma=0.5$ 时，求 $P\{1.8 < x < 2.9\}$ ， $P\{-3 < X\}$ ， $P\{|X - 2| > 1.5\}$ ；

(2) 若 $P\{X < x\} = 0.95$ ，求 x ；

(3) 分别绘制 $\sigma=0.2, 0.5, 0.9$ 时的概率密度函数图像。

其实现的 MATLAB 程序代码如下:

```
>> clear all;
p1=normcdf(2.9,2,0.5)-normcdf(1.8,0.5)      %计算  $P\{1.8 < X < 2.9\}$  的概率
p2=1-normcdf(-3,2,0.5)                      %计算  $P\{-3 < X\}$  的概率
p3=normcdf(0.5,2,0.5)+1-normcdf(3.5,2,0.5)
px=0.95;
x0=norminv(px,2,0.5)
x=-2:0.01:7;
y1=normpdf(x,2,0.2);
y2=normpdf(x,2,0.5);
y3=normpdf(x,2,0.9);
plot(x,y1,'r-',x,y2,'g-',x,y3);      %绘制概率密度函数图像
gtext('标准差为 0.2');
gtext('标准差为 0.5');
gtext('标准差为 0.9');
gtext('标准差为 1');
title('正态分布密度曲线');
```

运行程序, 输出如下, 效果如图 4-24 所示。

```
p1 =    0.0609
p2 =     1
p3 =    0.0027
x0 =    2.8224
```

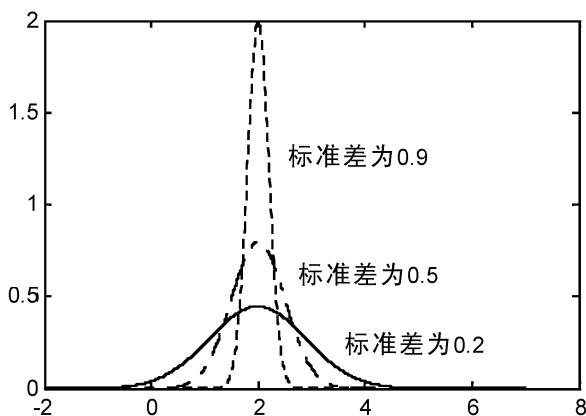


图 4-24 正态分布密度曲线效果

【例 4-81】泊松分布临界值示例。

```
>> poissinv(0.95,2)
ans =
     5
>> median_defects = poissinv(0.50,2)
median_defects =
     2
```

17. raylinv 函数

功能: Rayleigh 分布的临界值。其调用格式如下:

$X = \text{raylinv}(P,B)$

【例 4-82】Rayleigh 分布的临界值示例。

```
>> x = raylinv(0.9,1)
x =
    2.1460
```

18. tinv 函数

功能：T 分布的临界值。其调用格式如下：

$X = \text{tinv}(P,V)$

tinv 函数的逆累积分布公式如下：

$$x = F^{-1}(p | v) = \{x : F(x | v) = p\}$$

其中

$$p = F(x | v) = \int_{-\infty}^x \frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)} \frac{1}{\sqrt{v\pi}} \frac{1}{\left(1 + \frac{t^2}{v}\right)^{\frac{v+1}{2}}} dt$$

【例 4-83】T 分布的临界值示例。

```
>> percentile = tinva(0.99,1:6)
percentile =
    31.8205    6.9646    4.5407    3.7469    3.3649    3.1427
```

19. unidinv 函数

功能：离散平均分布临界值。其调用格式如下：

$X = \text{unidinv}(P,N)$

【例 4-84】离散平均分布临界值示例。

```
>> x = unidinv(0.7,20)
x =
    14
>> y = unidinv(0.7 + eps,20)
y =
    15
```

20. unifinv 函数

功能：连续均匀分布逆累积分布函数。其调用格式如下：

$X = \text{unifinv}(P,A,B)$

【例 4-85】连续均匀分布逆累积分布函数示例。

```
>> median_value = unifinv(0.5)
median_value =
    0.5000
>> percentile = unifinv(0.99,-1,1)
percentile =
    0.9800
```

4.2 随机数的使用

Galton 板实验是二项分布的一个经典的例子，而赌徒输光问题是人们早期进行概率问题研究的示例。下面分别对这两个例子进行介绍。

4.2.1 Galton 板实验

【例 4-86】模拟 Galton 板实验。

一个 8 级 Galton 板实验系统如图 4-25 所示。在图 4-25 (a) 中，当小球从顶部向下降落时，遇到第一层竖隔板，此时小球向左落下和向右落下的概率各占 $1/2$ ，即小球向左和向右运算的概率为 0.5；当小球继续下落遇到第二层竖隔板时，小球向左和向右的概率仍是 0.5，以后每一层情况都是如此（一个可能的过程如图 4-25 (b) 所示）。最后到了第 8 层底部，小球将落入底部 9 个槽中的其中一个。但是小球究竟落入哪个槽内的概率是不一样的。如果将这些槽编号为 1、2、3、4、5、6、7、8、9，计算小球落入 9 个槽中的概率。

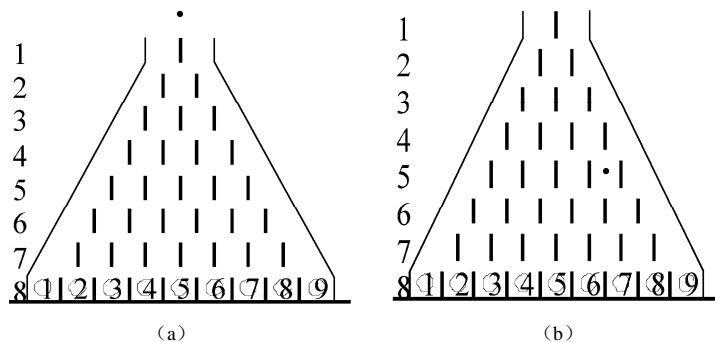


图 4-25 Galton 板实验

实现图 4-25 (a) 的 Galton_board1.m 的源程序代码如下：

```
>> %Galton 板实验
clear all;
xlim([-1,9]);
ylim([0,9]);
axis equal;
hold on;
L=0.8;
No=8;% 层数
for N=1:No;
    mN=No/2-[1+N]/2;
    for k=1:N;
        plot([mN+k]*[1,1],[0,0.7]+No-N,'k','linewidth',2);
        text(-0.9,No-N+0.3,num2str(N),'fontsize',14,...
            'fontname','times new roman');
    end
end
end
arg=linspace(0,pi*2,200);
```

```
col=[0.4,0.4,0.4];
fill(No/2+0.2*cos(arg),No+0.5+0.2*sin(arg),col,...
    'Edgecolor',col);
plot([No/2-1,No/2-1,-0.5,-0.5],...
    [No-1+0.7,No-1,0.7,0],'k');
plot([No/2+1,No/2+1,No+0.5,No+0.5],...
    [No-1+0.7,No-1,0.7,0],'k');
plot([-1,No+1],[0,0],'k','linewidth',2);
for k=1:No+1;
    text(k-1.1,0.4,num2str(k),'fontsize',14,...
        'fontname','times new roman');
    plot(k-1+0.24*cos(arg),0.4+0.24*sin(arg),'k');
end
set(gcf,'color','w');
axis off;
```

实现图 4-25 (b) 的 Galton_board2.m 的源程序代码如下:

```
>> %Galton 板实验
clear all;
xlim([-1,9]);
ylim([0,9]);
axis equal;
hold on;
L=0.8;
No=8;% 层数
for N=1:No;
    mN=No/2-[1+N]/2;
    for k=1:N;
        plot([mN+k]*[1,1],[0,0.7]+No-N,'k','linewidth',2);
        text(-0.9,No-N+0.3,num2str(N),'fontsize',14,...
            'fontname','times new roman');
    end
end
arg=linspace(0,pi*2,200);
col=[0.4,0.4,0.4];
fill(No/2+1.5+0.2*cos(arg),No/2-0.5+0.2*sin(arg),col,...
    'Edgecolor',col);
plot([No/2-1,No/2-1,-0.5,-0.5],...
    [No-1+0.7,No-1,0.7,0],'k');
plot([No/2+1,No/2+1,No+0.5,No+0.5],...
    [No-1+0.7,No-1,0.7,0],'k');
plot([-1,No+1],[0,0],'k','linewidth',2);
for k=1:No+1;
    text(k-1.1,0.4,num2str(k),'fontsize',14,...
        'fontname','times new roman');
    plot(k-1+0.24*cos(arg),0.4+0.24*sin(arg),'k');
end
set(gcf,'color','w');
```

axis off;

理论分析：这是一个经典的二项分布概率模型。考虑在第 k 层小球运动方向有两种可能，用 X_k 表示，则 X_k 服从两点概率分布。这里用 $X_k=1$ 表示竖隔板向右侧运动，用 $X_k=0$ 表示竖隔板向左侧运动，它们发生的可能性都是 50%。

最终位置 X 由下式决定：

$$X = \sum_k X_k$$

可知 X 满足二项分布，即 $X \sim B(8, 0.5)$ 。该二项分布的概率密度可用下面的程序代码获得：

bpdf=binopdf(0:8, 0.5);

模拟分析：用 MATLAB 模拟小球下落到 9 个槽内的概率分布。在各层中，用 0 和 1 分别表示向左和向右的运动。在小球下落到底层的槽内时，一个 8 位的二进制数就完全确定了。而所落入槽的编号应该是 8 位二进制数各数位按十进制相加的结果。重复模拟小球下降过程 8000 次，可以统计出小球落入各槽内的次数（即频数）。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
rand('state',0)
R=unidrnd(2,8,8000)-1;
test=sum(R);
h=hist(test,9);
bpdf=binopdf(0:8,8,0.5);
bpdf=bpdf/max(bpdf)*max(h);
b1=bar(1:9,[h;bpdf]);
set(b1(2),'facecolor','w');
set(b1(1),'facecolor',[0.4,0.4,0.4]);
```

运行程序，效果如图 4-26 所示。

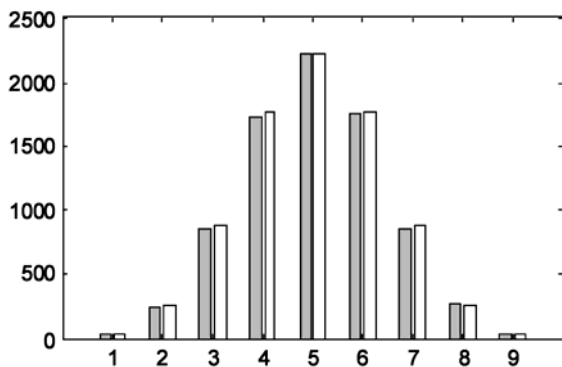


图 4-26 模拟的频率直方图比较

在图 4-26 中，灰色线条表示模拟值，而白色线条表示理论值（其中把概率理论值的最大值调整为模拟值的最大值以便于比较），可以发现二者符合得很好。

4.2.2 输赢问题

【例 4-87】赌徒输赢问题。

两个赌徒甲、乙将进行一系列赌博。在每一局中甲获胜的概率为 p ，而乙获胜的概率为 q ， $p+q=1$ 。在每一局中，失败者都要付一元钱给胜利者。在开始时甲拥有资本 a 元，而乙拥有资本 b 元，两个赌徒直到甲输光或乙输光而停止赌博。求甲输光所有钱的概率。

通过理论分析可知甲输光所有钱的概率为

$$P = \begin{cases} \frac{b}{a+b} & (p = \frac{1}{2} \text{ 时}) \\ \frac{1-g^b}{1-g^{a+b}}, g = \frac{p}{q} & (p \neq \frac{1}{2} \text{ 时}) \end{cases}$$

模拟赌博过程的思想：在每一次模拟中，随机地产生 1 个数，如果该数小于 p ，说明赌博甲胜，相应甲得到一元钱，而乙付一元钱；反之甲拿出一元钱给乙。这里对甲的赌资 a 、乙的赌资 b 、甲赢得的概率 p 取不同数值进行 10000 次赌博过程模拟。其实现的 MATLAB 程序代码如下：

```
>> clear all;
a=10;      %假设甲的赌资为 10 元
b=3;       %假设乙的赌资为 3 元
p=0.55;    %甲赢得的概率为 55%
S=0;       % 计数器置 0
N=10000;   % 模拟的次数
m=6;
rand('state',m); % 设置随机数状态
for k=1:N;
    at=a;
    bt=b;
    while at>0.5&bt>0.5;
        r=[(rand<p)-0.5]*2; % 算输赢
        at=at+r;
        bt=bt-r;
    end
    S=S+(at<0.5); % 累加甲输的次数
end
P=S/N % 模拟的概率值
g=p/[1-p];
Po=[1-g^b]/[1-g^(a+b)] % 概率的理论值
```

运行程序，输出如下：

```
P =
    0.0638
Po =
    0.0656
```

4.3 统计量的数字特征

4.3.1 数学期望与均值

数学期望是概率统计中的重要概念，在给定的一组样本 $x=[x_1, x_2, \dots, x_i]$ ，简单的数学期望可

由下式计算:

$$E(x) = \frac{1}{n} \sum_{i=1}^n x_i$$

此时期望值等于各个元素的算术平均值。

使用 `mean` 函数求解一组样本数据 x 的算术平均值。其调用格式如下:

`M = mean(A)`

`M = mean(A,dim)`

【例 4-88】求算术平均值示例。

```
>> A = [1 2 3; 3 3 6; 4 6 8; 4 7 7];
mean(A)
ans =
    3.0000    4.5000    6.0000
>> mean(A,2)
ans =
     2
     4
     6
     6
```

在 MATLAB 中, 还提供了其他几个求平均数的函数。

- (1) `nanmean` 函数: 求随机变量的算术平均值。
- (2) `geomean` 函数: 求随机变量的几何平均值。
- (3) `harmmean` 函数: 求随机变量的调和平均值。
- (4) `median` 函数: 求随机变量的中位数。
- (5) `nanmedian` 函数: 返回样本数据 X 中除 NaN 外的中位数。

【例 4-89】其他几个求平均数的函数示例。

```
>> dates = {'14-01-2010'; '15-01-2010'; '16-01-2010'};
f = fints(dates, magic(3));
f.series1(1) = nan;
f.series2(3) = nan;
f.series3(2) = nan;
nmean = nanmean(f)           %随机变量的算术平均值
nmean =
    3.5000    3.0000    4.0000
>> dates = {'14-01-2010'; '15-01-2010'; '16-01-2010'; '17-01-2010'};
f = fints(dates, magic(4));
f.series1(1) = nan;
f.series2(2) = nan;
f.series3([1 3]) = nan;
nmedian = nanmedian(f)       %返回样本数据 X 中除 NaN 外的中位数
nmedian =
    5.0000    7.0000   12.5000   10.0000
>> x = exprnd(1,10,6);
geometric = geomean(x)       %随机变量的几何平均值
geometric =
    0.7950    0.5377    0.5139    0.3620    0.8460    0.8238
```

```
>> x = exprnd(1,10,6);    %随机变量的调和平均值
harmonic = harmmean(x)
harmonic =
    0.1798    0.2589    0.0587    0.6157    0.1907    0.2568
>> A = [1 2 4 4; 3 4 6 6; 5 6 8 8; 5 6 8 8];    %随机变量的中位数
median(A)
ans =
     4     5     7     7
```

4.3.2 数据比较

在给定的数据中，下面一些函数可以对数据进行排序以及求数据的最大值和最小值。

- (1) max 函数：求随机变量的最大元素。
- (2) min 函数：求随机变量的最小元素。
- (3) nanmax 函数：求随机变量忽略 NaN 的最大值元素。
- (4) nanmin 函数：求随机变量忽略 NaN 的最小值元素。
- (5) sort 函数：用于进行数据由小到大的排序。
- (6) sortrows 函数：按首行进行数据的排序。
- (7) range 函数：求解数据中最大值与最小值之差。
- (8) mad 函数：求随机变量的绝对差分平均值。
- (9) median 函数：求随机变量的中值。

【例 4-90】数据比较示例。

```
>> A=[9 8 11 0 7 3 2 17 5 4];
>> max(A)    %随机变量的最大元素
ans =    17
>> min(A)    %随机变量的最小元素
ans =     0
>> nanmax(A) %随机变量忽略 NaN 的最大值元素
ans =    17
>> nanmin(A) %随机变量忽略 NaN 的最小值元素
ans =     0
>> sort(A)   %进行数据由小到大的排序
ans =
     0     2     3     4     5     7     8     9    11    17
>> sortrows(A) %按首行进行数据的排序
ans =
     9     8    11     0     7     3     2    17     5     4
>> range(A)  %求解数据中最大值与最小值之差
ans =    17
>> mad(A)    %求随机变量的绝对差分平均值
ans =    3.8000
>> median(A) %求随机变量的中值
ans =     6
```

4.3.3 方差和标准差

方差的计算是概率统计中的一项重要内容，它可以表征随机变量与其他均值的偏离程度。

方差定义为 $D(x) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$ ，使用 `var` 函数来求解样本数据的方差，其调用格式如下：

下：

V = var(X)：返回样本数据的方差，当 **X** 为向量时，该命令返回 **X** 向量的样本方差；当 **X** 为矩阵时，返回由 **X** 矩阵的列向量的样本方差构成的行向量。

V = var(X,1)：返回向量（矩阵）**X** 的简单方阵（即置前因子为 1/n 的方差）。

V = var(X,w)：返回以 **w** 为权重的向量（矩阵）**X** 的方差。

V = var(X,w,dim)：返回给出 **X** 的维数 **dim** 内的方差。

样本的标准差即为 $\sigma(x) = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$ ，使用 `std` 函数用来求解样本数据的标准差。其调用格式如下：

s = std(X)：返回向量（矩阵）**X** 的样本标准差（置前因子为 1/(n-1)），即

$$\text{std} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

s = std(X,flag)：返回向量（矩阵）**X** 的标准差（置前因子为 1/n）。

s = std(X,1,dim)：返回向量（矩阵）**X** 的标准差（置前因子为 1/n）。

s = std(X,flag,dim)：返回向量（矩阵）中维数为 **dim** 的标准差值，其中 **flag=0** 时，置前因子为 1/(n-1)；否则置前因子为 1/n。

【例 4-91】求解样本方差。

```
>> W=[0.1 0.2 0.3 0.4];
>> B=magic(4)      %魔方矩阵
B =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> var(W)
ans =
    0.0167

>> var(B)
ans =
    29.6667    27.0000    27.0000    29.6667

>> var(B,1)
ans =
    22.2500    20.2500    20.2500    22.2500

>> var(B,W)
ans =
    13.6900    15.6900    19.6900    25.6900

>> std(B)
ans =
     5.4467     5.1962     5.1962     5.4467
```



```
>> std(B,0,1)
ans =
    5.4467    5.1962    5.1962    5.4467
>> std(B,1,1)
ans =
    4.7170    4.5000    4.5000    4.7170
>> std(B,1,2)
ans =
    6.1033
    2.2913
    2.2913
    6.1033
```

【例 4-92】正态分布标准差意义的图形表示。

```
%正态分布参数设定
mu=3;
sigma=0.5;
x=mu+sigma*[-3:-1,1:3];
yf=normcdf(x,mu,sigma);
%计算  $P(\mu - k \cdot \sigma \leq x \leq \mu + k \cdot \sigma)$ 
P=[yf(4)-yf(3),yf(5)-yf(2),yf(6)-yf(1)];
%计算概率密度函数，以供图形表示
xd=1:0.1:5;yd=normpdf(xd,mu,sigma);
%为各区域填色而进行的计算
for k=1:3
    xx{k}=x(4-k):sigma/10:x(3+k);    %用元胞数组存放采样数不同的数据
    yy{k}=normpdf(xx{k},mu,sigma);    %用元胞数组存放采样数不同的数据
end
subplot(1,3,1),plot(xd,yd,'b');
hold on
fill([x(3),xx{1},x(4)],[0,yy{1},0],'g')
text(mu-0.5*sigma,0.3,num2str(P(1))),hold off
subplot(1,3,2),plot(xd,yd,'b');
hold on
fill([x(2),xx{2},x(5)],[0,yy{2},0],'g')
text(mu-0.5*sigma,0.3,num2str(P(2))),hold off
subplot(1,3,3),plot(xd,yd,'b');
hold on
fill([x(1),xx{3},x(6)],[0,yy{3},0],'g')
text(mu-0.5*sigma,0.3,num2str(P(3))),hold off
```

运行程序，效果如图 4-27 所示。

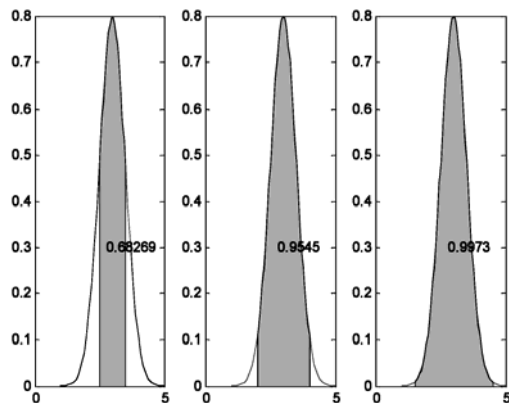


图 4-27 均值两侧一、二、三倍标准差之间的概率

4.3.4 累积和累和

求向量和矩阵的累积和累和运算可由以下函数来实现。

- (1) **sum** 函数：求向量和矩阵的累和。
- (2) **nansum**：忽略 NaN 求向量和矩阵的累和。
- (3) **cumsum**：求此元素位置以前的元素和。
- (4) **cumtrapz**：梯形累和函数。
- (5) **cumprod**：求当前元素与所有前面位置的元素积。

【例 4-93】累积和累和示例。

```
>> M = magic(3)
M =
     8     1     6
     3     5     7
     4     9     2

>> sum(M)
ans =
    15    15    15

>> nansum(M)
ans =
    15    15    15

>> cumsum(M)
ans =
     8     1     6
    11     6    13
    15    15    15

>> cumtrapz(M)
ans =
     0     0     0
    5.5000    3.0000    6.5000
    9.0000   10.0000   11.0000

>> cumprod(M)
ans =
```

8	1	6
24	5	42
96	45	84

4.3.5 协方差与相关系数

对于二维随机变量 (X, Y) ，除了讨论 X 和 Y 的数学期望和方差以外，还需要讨论描述 X 和 Y 之间的相互关系的数学特征，下面将介绍协方差和相关系数。

协方差： $\text{cov}(x, y) = E\{[x - E(x)][y - E(y)]\}$

协方差矩阵： $\text{cov}(x, y) = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$

其中：

$$C_{11} = E\{[x - E(x)]^2\}$$

$$C_{12} = E\{[x - E(x)][y - E(y)]\}$$

$$C_{21} = E\{[y - E(y)][x - E(x)]\}$$

$$C_{22} = E\{[y - E(y)]^2\}$$

相关系数：

$$\text{cor}(x, y) = \frac{\text{cov}(x, y)}{\sqrt{D(x)D(y)}}$$

在 MATLAB 中提供了 `cov` 函数用来求解协方差。其调用格式如下：

`z1=cov(x)`

`z2=cov(x,y)`

`z3= cov(x,y,1)`

当 x 为向量时， $z1$ 返回值为向量 x 的协方差；当 x 为矩阵时， $z1$ 返回值为矩阵 x 的协方差矩阵，该协方差矩阵的对角线元素是 x 的各列方差； $z2$ 返回向量 x, y 的协方差矩阵， x, y 的维数必须相同； $z3$ 返回向量 x, y 的协方差矩阵，置前因子为 $1/n$ 。

在 MATLAB 中提供了 `corrcoef` 函数用来求解相关系数。其调用格式如下：

`R = corrcoef(X)`：返回矩阵 x 的列向量的相关系数矩阵。

`R = corrcoef(x,y)`：返回列向量 x 与 y 的相关系数。

【例 4-94】协方差与相关系数示例。

```
>> a=rand(3,3)           %随机矩阵
a =
    0.8654    0.6396    0.1089
    0.9798    0.7050    0.7251
    0.9509    0.6101    0.5314
>> b=magic(3)           %魔方矩阵
b =
     8     1     6
     3     5     7
     4     9     2
>> cov(a)
```

```
ans =
    0.0035    0.0013    0.0187
    0.0013    0.0024    0.0077
    0.0187    0.0077    0.0993
>> cov(a,b)
ans =
    0.0694   -0.0584
   -0.0584    7.5000
>> corrcoef(a)
ans =
    1.0000    0.4438    0.9978
    0.4438    1.0000    0.5026
    0.9978    0.5026    1.0000
>> corrcoef(a,b)
ans =
    1.0000   -0.0810
   -0.0810    1.0000
```

4.3.6 偏斜度和峰度

为描述随机变量分布的形状与对称形状或正态分布的偏离程度，引入了特征量的偏斜度和峰度。

1. 偏斜度

偏斜度的定义：

$$v_1 = E \left[\left(\frac{x - E(x)}{\sqrt{D(x)}} \right)^3 \right]$$

此函数表征分布形状偏斜对称的程度，若 $v_1=0$ ，则可以认为分布是对称的；若 $v_1>0$ ，则称为右偏态，此时位于均值右边的峰值比位于左边的值多一些；若 $v_1<0$ ，则称为左偏态，即位于均值左边的值比位于右边的值多一些。

此函数在 MATLAB 中的功能由 `skewness` 函数来实现。其调用格式如下：

`y = skewness(X)`：若 `X` 为向量，则函数返回此向量的偏斜度；若 `X` 为矩阵，则返回矩阵列向量的偏斜度行向量。

`y = skewness(X,flag)`：指定是否纠正偏离（`flag=0` 时，是；`flag=1` 时，否）后，再返回偏斜度。

【例 4-95】偏斜度示例。

```
>> X = randn([5 4])
X =
   -0.4326    1.1909   -0.1867    0.1139
   -1.6656    1.1892    0.7258    1.0668
    0.1253   -0.0376   -0.5883    0.0593
    0.2877    0.3273    2.1832   -0.0956
   -1.1465    0.1746   -0.1364   -0.8323
>> y = skewness(X)
y =
```

```
-0.2752    0.2569    0.9066    0.2632
>> y = skewness(X,0)
y =
-0.4102    0.3829    1.3514    0.3923
```

2. 峰度

峰度的定义：

$$v_2 = E \left[\left(\frac{x - E(x)}{\sqrt{D(x)}} \right)^4 \right]$$

若 $v_2 > 0$ ，表示分布有着沉重的“尾巴”，即数据中含有较多偏离均值的数据。对于正态分布， $v_2 = 0$ ，故 v_2 的值也可看成是数据偏离正态分布的尺度。在 MATLAB 中此功能由 kurtosis 来实现，其调用格式如下：

k = kurtosis(X)：若 X 为向量，则函数返回此向量的峰度；若 X 为矩阵，则返回矩阵列向量的峰度行向量。

k = kurtosis(X,flag)：指定是否纠正偏离（flag=0 时，是；flag=1 时，否）后，再返回峰度。

k = kurtosis(X,flag,dim)：返回给出 X 的维数 dim 的峰度。

【例 4-96】有 12 名学生体重(单位为 kg)为 78.5, 69.2, 48.7, 66.9, 80, 79.1, 81, 69.5, 68.2, 55.9, 66.7, 78。计算此 12 名学生体重的均值、标准差、偏斜度和峰度。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
w=[78.5, 69.2, 48.7, 66.9, 80, 79.1, 81, 69.5, 68.2, 55.9, 66.7, 78];
m1=mean(w)
s1=std(w)
sk1=skewness(w)
ku1=kurtosis(w)
```

运行程序，输出如下：

```
m1 = 70.1417
s1 = 10.0767
sk1 = -0.8169
ku1 = 2.8014
```

4.4 数据的属性与处理方法

4.4.1 评价指标矩阵与指标的无量纲化

评价指标通常分为效益型、成本型、固定型等，效益型指标值越大越好、成本型指标值越小越好、固定型指标值既不能太大也不能太小为好。

对方案进行综合评价，必须统一评价指标的属性，即进行指标的无量纲化处理。常见的处理方法有极差变换、线性比例变换、向量归一化、标准样本变换、等效系数法等。下面说明极差变换及线性比例变换，其他处理方法请读者参考其他相关书籍。

设 n 个决策方案的集合为 $A = \{A_1^T, A_2^T, \dots, A_n^T\}$ ，其中 $A_i^T = (a_{i1}, a_{i2}, \dots, a_{im})$ 是第 i 个方案关于

第 m 项评价指标的指标值向量。记为 n 个方案关于 m 项评价指标的指标矩阵为

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

其中: a_{ij} 表示第 i 个方案关于第 j 项评价因素的指标值。

下面用 I_1 、 I_2 、 I_3 分别表示效益型、成本型和固定型指标集合。

1. 运用极差变换法建立无量纲的效益型矩阵 B 与成本型矩阵 C

(1) 效益型矩阵

$$B = (b_{ij})_{n \times m}, \quad b_{ij} = \begin{cases} \frac{(a_{ij} - \min_j a_{ij})}{(\max_j a_{ij} - \min_j a_{ij})} & (a_{ij} \in I_1) \\ \frac{(\max_j a_{ij} - a_{ij})}{(\max_j a_{ij} - \min_j a_{ij})} & (a_{ij} \in I_2) \\ \frac{(\max_j |a_{ij} - \alpha_j| - |a_{ij} - \alpha_j|)}{\max_j |a_{ij} - \alpha_j| - \min_j |a_{ij} - \alpha_j|} & (a_{ij} \in I_3) \end{cases}$$

式中: α_j 为第 j 项指标的适度数值。

显然指标经过极差变换后, 均有 $0 \leq b_{ij} \leq 1$, 且各指标下最好结果的属性值 $b_{ij}=1$, 最坏结果的属性值 $b_{ij}=0$ 。指标变换前后的属性值成比例。

(2) 成本型矩阵

$$C = (c_{ij})_{n \times m}, \quad c_{ij} = \begin{cases} \frac{(\max_j a_{ij} - a_{ij})}{(\max_j a_{ij} - \min_j a_{ij})} & (a_{ij} \in I_1) \\ \frac{(a_{ij} - \min_j a_{ij})}{(\max_j a_{ij} - \min_j a_{ij})} & (a_{ij} \in I_2) \\ \frac{|a_{ij} - \alpha_j| - \min_j |a_{ij} - \alpha_j|}{\max_j |a_{ij} - \alpha_j| - \min_j |a_{ij} - \alpha_j|} & (a_{ij} \in I_3) \end{cases}$$

式中: α_j 为第 j 项指标的适度数值。

显然指标经过极差变换后, 均有 $0 \leq c_{ij} \leq 1$, 且各指标下最坏结果的属性值 $c_{ij}=1$, 最好结果的属性值 $c_{ij}=0$ 。

2. 运用线性比例变换法建立无量纲的效益型矩阵 D 与成本型矩阵 E

(1) 效益型矩阵

$$D = (d_{ij})_{n \times m}, \quad d_{ij} = \begin{cases} \frac{a_{ij}}{\max_j a_{ij}} & (a_{ij} \in I_1) \\ \frac{\min_j a_{ij}}{a_{ij}} & (a_{ij} \in I_2) \\ \frac{\min_j |a_{ij} - \alpha_j|}{|a_{ij} - \alpha_j|} & (a_{ij} \in I_3) \end{cases}$$

(2) 成本型矩阵

$$E = (e_{ij})_{n \times m}, \quad e_{ij} = \begin{cases} \frac{\min_j a_{ij}}{a_{ij}} & (a_{ij} \in I_1) \\ \frac{a_{ij}}{\max_j a_{ij}} & (a_{ij} \in I_2) \\ \frac{|a_{ij} - \alpha_j|}{\max_j |a_{ij} - \alpha_j|} & (a_{ij} \in I_3) \end{cases}$$

式中: α_j 为第 j 项指标的适度数值。显然指标变换前后的属性值成比例。

4.4.2 客观性权向量建立的方法

指标权重的合理确定是综合评价结果是否可信的一个核心问题, 确定权重系数的途径有三类: 一是主观赋权法, 二是客观赋权法, 三是主客观结合赋权法。这里主要介绍 3 种客观赋权法: 一是变量系数法, 二是夹角余弦法, 三是熵值法。

1. 变量系数法

对无量纲的理想值矩阵 $H=(h_{ij})_{n \times m}$, 计算各列向量的变异系数 $v_j = \frac{s_j}{h_j} (j=1, 2, \dots, m)$, 其中 s_j

是第 j 列的标准差, 然后将其归一化就得到权向量。

2. 夹角余弦法

设无量纲的效益型矩阵为 $B=(b_{ij})_{n \times m}$, 则可得到各方案与理想最佳和最劣方案的相对偏差矩阵为

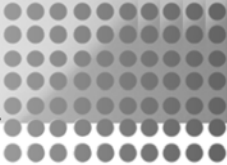
$$U=(u_{ij})_{n \times m}, \quad V=(v_{ij})_{n \times m}$$

式中: $u_{ij} = \frac{\max_j b_{ij} - b_{ij}}{\max_j b_{ij} - \min_j b_{ij}}$, $v_{ij} = \frac{b_{ij} - \min_j b_{ij}}{\max_j b_{ij} - \min_j b_{ij}}$, 再计算 U, V 对应列向量的夹角余弦得到初

始权重, 归一化后得到客观性权向量。

3. 熵值法

(1) 对决策矩阵 $X=(x_{ij})_{n \times m}$ 作标准化处理, 得到标准化矩阵 $Y=(y_{ij})_{n \times m}$, 并进行归一化处理得



$$p_{ij} = \frac{y_{ij}}{\sum_{i=1}^m y_{ij}} \quad (1 \leq i \leq m, 1 \leq j \leq n)$$

(2) 计算第 j 个指标的熵值:

$$e_j = -k \cdot \sum_{i=1}^m p_{ij} \ln p_{ij} \quad (1 \leq j \leq n, k > 0, e_j \geq 0)$$

(3) 计算第 j 个指标的差异系数。差异系数定义为

$$g_j = 1 - e_j \quad (1 \leq j \leq n)$$

显然, 对于第 j 个指标, 指标值的差异越大, 对方案评价的作用越大, 熵值越小; 反之, 差异越小, 对方案评价的作用越小, 熵值就越大。

(4) 确定指标权重。第 j 个指标的权重为

$$w_j = \frac{g_j}{\sum_{j=1}^n g_j} \quad (1 \leq j \leq n)$$

4.4.3 综合评价的步骤

综合评价一般按以下步骤进行:

- (1) 确定综合评价指标体系, 这是综合评价的基础和依据。由于以下步骤的操作较为确定, 因此, 指标的选择往往是综合评价科学性的关键。
- (2) 搜集数据并进行同度量处理, 以消除量纲的影响。
- (3) 确定指标权重。由于参评指标的重要性是不同的, 所以要根据指标的重要性大小进行加权处理。
- (4) 对经过处理后的指标值(变量值)进行汇总, 计算综合评价指数或综合评价分值。
- (5) 根据综合评价指数或综合评价分值对参评单位进行排序。

4.4.4 数据的属性与处理方法示例

1. 湖泊的水质评价模型

【例 4-97】近年来我国淡水湖水质富营养化的污染日趋严重, 正确评价湖水的水质情况, 有利于今后开展对湖水的污染治理和保护工作。表 4-1 所列为我国 5 个湖泊的评价参数实测数据, 表 4-2 所列湖泊水质评价标准, 试建立模型对我国的 5 个湖泊的水质进行评价, 以确定各湖水水质等级。

表 4-1 全国 5 个主要湖泊的实测数据

指标 湖泊	总磷/(mg/L)	耗氧量/(mg/L)	透明度/m	总氮/(mg/L)
杭州西湖	130	10.30	0.35	2.76
武汉东湖	105	10.70	0.40	2.0
青海湖	20	1.4	4.5	0.22



续表

<div>指标 湖泊</div>	总磷/ (mg/L)	耗氧量/ (mg/L)	透明度/m	总氮/ (mg/L)
巢湖	30	6.26	0.25	1.67
滇湖	20	10.13	0.50	0.23

表 4-2 湖泊水质评价标准

<div>参数值 指 标</div>	极贫营养	贫营养	中营养	富营养	极富营养
总磷/ (mg/L)	<1	4	23	110	>660
耗氧量/ (mg/L)	<0.09	0.36	1.80	7.10	>27.1
透明度/m	>37	12	2.4	0.55	<0.17
总氮/ (mg/L)	<0.02	0.06	0.31	1.20	>4.6

分析：这是一个多指标评价问题，影响水质的指标有总磷、耗氧量、透明度、总氮，其中透明度是效益型指标，其余的是成本型指标。

(1) 由表 3-7 和表 3-8，建立实测指标数据矩阵：

$$X = (x_{ij})_{5 \times 4} = \begin{bmatrix} 130 & 10.3 & 0.35 & 2.76 \\ 105 & 10.7 & 0.4 & 2.0 \\ 20 & 1.4 & 4.5 & 0.22 \\ 30 & 6.26 & 0.25 & 1.67 \\ 20 & 10.13 & 0.5 & 0.23 \end{bmatrix}$$

建立等级标准矩阵：

$$Y = (y_{ij})_{4 \times 5} = \begin{bmatrix} 1 & 4 & 23 & 110 & 660 \\ 0.09 & 0.36 & 1.8 & 7.10 & 27.1 \\ 37 & 12 & 2.4 & 0.55 & 0.17 \\ 0.02 & 0.06 & 0.31 & 1.20 & 4.60 \end{bmatrix}$$

运用线性比例变换法将 X 无量纲化得效益型指标矩阵：

$$A = (a_{ij})_{5 \times 4}, \text{ 其中 } a_{ij} = \begin{cases} \frac{x_{ij}}{\max_j x_{ij}} & (j \neq 3) \\ \frac{\min_j x_{ij}}{x_{ij}} & (j = 3) \end{cases}$$

将 Y 无量纲化得效益型等级标准矩阵：

$$B = (b_{kt})_{4 \times 5}, \text{ 其中 } b_{kt} = \begin{cases} \frac{y_{kt}}{\max_k y_{kt}} & (k \neq 3) \\ \frac{\min_k y_{kt}}{y_{kt}} & (k = 3) \end{cases}$$

(2) 计算评价指标的权重。运用变异系数法由矩阵 B 来确定各指标的权重，具体过程为首先计算 B 各行向量的均值与标准差：

$$\mu_i = \frac{1}{5} \sum_{j=1}^5 b_{ij}, \quad s_i = \sqrt{\frac{\sum_{j=1}^5 (b_{ij} - \mu_i)^2}{4}} \quad (i=1,2,3,4)$$

再计算变异系数 $v_i = \frac{s_i}{\mu_i} (i=1,2,3,4)$ ，最后对变异系数归一化得到各指标的权向量为

$w=(w_1, w_2, w_3, w_4)$ 。

(3) 建立水质的等级评价模型。利用欧氏距离和绝对距离进行建模，计算 A 中各行向量到 B 中各列向量的欧氏距离 d_{ij} ：

$$d_{ij} = \sqrt{\sum_{k=1}^4 (a_{ik} - b_{kj})^2} \quad (i=1,2,3,4,5; j=1,2,3,4,5)$$

若 $d_{ik} = \min_{1 \leq j \leq 5} \{d_{ij}\}$ ，则第 i 个湖泊属于第 k 级 ($i=1,2,3,4,5$)。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
% 输入原始数据
X=[130 10.30 0.35 2.76;105 10.70 0.40 2.0;20 1.4 4.5 0.22;30 6.26 0.25 1.67;20 10.13 0.50 0.23];
Y=[1 4 23 110 660;0.09 0.36 1.80 7.10 27.1;37 12 2.4 0.55 0.17;0.02 0.06 0.31 1.20 4.6];
%计算无量纲化的指标矩阵
A=[X(:,1)./max(X(:,1)),X(:,2)./max(X(:,2)),min(X(:,3))./X(:,1),X(:,4)./max(X(:,4))]
A =
    1.0000    0.9626    0.0019    1.0000
    0.8077    1.0000    0.0024    0.7246
    0.1538    0.1308    0.0125    0.0797
    0.2308    0.5850    0.0083    0.6051
    0.1538    0.9467    0.0125    0.0833
>> B=[Y(1,:)./max(Y(1,:));Y(2,:)./max(Y(2,:));min(Y(3,:))./Y(3,:);Y(4,:)./max(Y(4,:))]
B =
    0.0015    0.0061    0.0348    0.1667    1.0000
    0.0075    0.0300    0.1500    0.5917    2.2583
    0.0046    0.0142    0.0708    0.3091    1.0000
    0.0043    0.0130    0.0674    0.2609    1.0000
>> %运用变异系数法由矩阵 B 来确定各指标的权重
b=B';
t=std(b)./mean(b);
```

```
w=t/sum(t)
w =
    0.2767    0.2444    0.2347    0.2442
>> %计算绝对距离
jd=dist(A,B)
jd =
    1.7031    1.6828    1.5705    1.2136    1.6356
    1.4676    1.4457    1.3253    0.9417    1.6406
    0.2101    0.1909    0.1345    0.5773    2.6579
    0.8643    0.8421    0.7216    0.4616    2.1286
    0.9548    0.9312    0.8078    0.4957    2.0620
>> %计算欧氏距离
mjd=mandist(A,B)
mjd =
    2.9519    2.9258    2.7793    2.2506    2.2938
    2.5212    2.4950    2.3485    1.8198    2.7236
    0.3589    0.3170    0.2088    0.9514    4.8814
    1.4113    1.3776    1.2311    0.7157    3.8291
    1.1785    1.1365    0.9900    0.8420    4.0619
```

说明:

(1) 由权重向量 $w=[0.2767 \ 0.2444 \ 0.2347 \ 0.2442]$ 知, 4 项指标在湖泊水质富营养化中的权重依次是总磷最大、耗氧量与总氮次之、透明度最小, 换个角度可以说总磷的变化对湖泊水质富营养化作用最大。

(2) 由绝对距离结果可知, 杭州西湖、武汉东湖水质评为 5 级, 青海湖水质评为 3 级, 巢湖、滇池水质评为 4 级。由欧氏距离结果可知, 杭州西湖、武汉东湖水质评为 5 级, 青海湖、滇池水质评为 3 级, 巢湖评为 4 级。两种距离的评估结果有一点差别, 这说明方法还值得改进。

2. 经济效益综合评价模型

【例 4-98】设北京、上海、天津和昆明 4 个城市的 6 项经济效益指标统计数据见表 4-3, 试建立综合评价模型, 对这 4 个地区经济效益进行评价。

表 4-3 经济效益指标统计数据

指标 地区	资金 利润率/%	销售 利润率/%	全员劳动 生产率	综合 能耗	物耗	技改占固定 资产投资比率/%
北京	29.09	24.05	1.94	4.55	67.40	67.60
上海	36.97	22.90	2.60	2.43	67.90	54.55
天津	29.13	20.40	1.97	3.60	68.70	64.00
昆明	23.92	27.20	1.17	7.92	58.10	55.20

分析: 在表 4-3 的 6 项指标中, 综合能耗和物耗是成本型指标, 其余指标是效益型指标。

(1) 建立指标矩阵 $X=(x_{ij})_{4 \times 6}$, 其中 x_{ij} 表示第 i 个城市第 j 个指标的值。用极差变换法将 X 无量纲化得效益型矩阵 B 与成本型矩阵 D ; 运用线性比例变换法将 X 无量纲化得效益型矩阵 C 与成本型矩阵 E 。

(2) 运用夹角余弦法建立客观性权重向量。首先由指标矩阵 X 得到各方案与理想最佳和最

劣方案的相对偏差矩阵 R 与矩阵 T , 其次求出 R 与 T 两矩阵的对应列向量的夹角余弦, 并作为初始权重, 归一化后得到客观性权向量 w 。

(3) 计算综合评价值。由矩阵 B 可得 i 个城市的综合评价得分 $H_i = \sum_{j=1}^6 b_{ij} w_j$, 且 H_i 值越大越

好。同理, D 、 C 与 E 得第 i 个城市的综合评价得分分别是 $F_i = \sum_{j=1}^6 d_{ij} w_j$ 、 $h_i = \sum_{j=1}^6 c_{ij} w_j$ 与

$$f_i = \sum_{j=1}^6 e_{ij} w_j \quad (i=1,2,3,4,5)。$$

其实现的 MATLAB 程序代码如下:

```
>> clear all;
A=[29.09 24.05 1.94 4.55 67.40 67.60;36.97 22.90 2.60 2.43 67.90 54.55;...
 29.13 20.40 1.97 3.60 68.70 64.00;23.92 27.20 1.17 7.92 58.10 55.20];
%运用极差法建立无量纲的效益型矩阵 B
B=[(A(:,1:3)-ones(4,1)*min(A(:,1:3)))/(ones(4,1)*max(A(:,1:3))-min(A(:,1:3))),...
  (ones(4,1)*max(A(:,4:5))-A(:,4:5))/(ones(4,1)*max(A(:,4:5))-min(A(:,4:5))),...
  (A(:,6)-min(A(:,6)))/(ones(4,1)*max(A(:,6))-min(A(:,6)))];
B =
    0.3962    0.5368    0.5385    0.6138    0.1226    1.0000
    1.0000    0.3676    1.0000    1.0000    0.0755         0
    0.3992         0    0.5594    0.7869         0    0.7241
         0    1.0000         0         0    1.0000    0.0498
>> %运用线性比例变换法建立无量纲的效益型矩阵 D
D=[(A(:,1:3)-min(A(:,1:3)))/(max(A(:,1:3))-min(A(:,1:3))),...
  (max(A(:,4:5))-A(:,4:5))/(max(A(:,4:5))-min(A(:,4:5))),...
  (A(:,6)-min(A(:,6)))/(max(A(:,6))-min(A(:,6)))];
D =
    0.7869    0.8842    0.7462    0.5341    0.8620    1.0000
    1.0000    0.8419    1.0000    1.0000    0.8557    0.8070
    0.7879    0.7500    0.7577    0.6750    0.8457    0.9467
    0.6470    1.0000    0.4500    0.3068    1.0000    0.8166
>> %理想最佳和最劣方案向量 U 与 V
U=[max(A(:,1:3)),min(A(:,4:5)),max(A(:,6))]
V=[min(A(:,1:3)),max(A(:,4:5)),min(A(:,6))]
U =
    36.9700    27.2000     2.6000     2.4300    58.1000    67.6000
V =
    23.9200    20.4000     1.1700     7.9200    68.7000    54.5500
>> %计算相对偏差矩阵 R 与 T
R=abs(A-ones(4,1)*U)/(ones(4,1)*range(A))
T=abs(A-ones(4,1)*V)/(ones(4,1)*range(A))
R =
    0.6038    0.4632    0.4615    0.3862    0.8774         0
         0    0.6324         0         0    0.9245    1.0000
    0.6008    1.0000    0.4406    0.2131    1.0000    0.2759
    1.0000         0    1.0000    1.0000         0    0.9502
```

```
T =
    0.3962    0.5368    0.5385    0.6138    0.1226    1.0000
    1.0000    0.3676    1.0000    1.0000    0.0755         0
    0.3992         0    0.5594    0.7869         0    0.7241
         0    1.0000         0         0    1.0000    0.0498
```

```
>> %运用夹角余弦法建立权重向量 w
```

```
r=normc(R);
```

```
t=normc(T);
```

```
w=sum((r.*t))/sum(sum(r.*t))
```

```
w =
```

```
    0.2151    0.2148    0.2231    0.1774    0.0733    0.0962
```

```
>> %计算综合评价值
```

```
H=B*(w')
```

```
F=D*(w')
```

```
H =
```

```
    0.5348
```

```
    0.7001
```

```
    0.4200
```

```
    0.2930
```

```
F =
```

```
    0.7799
```

```
    0.9369
```

```
    0.7725
```

```
    0.6607
```

说明:

(1) 权重向量 $w=[0.2151 \ 0.2148 \ 0.2231 \ 0.1774 \ 0.0733 \ 0.0962]$, 从中可知物耗指标权重最小, 技改占固定资产投资比率指标权重次之, 全员劳动生产率权重最大。

(2) 求无量纲的成本型矩阵 C 与 E 的程序没有写出, 请读者补充, 对应的综合评价值的程序也请读者补充。

(3) 综合评价结果与排序从上面的显示结果可以看出, 对于两类不同的效益型矩阵和成本型矩阵, 综合评估的结果完全一样, 表明此方法具有较高的可靠性。

第 5 章 统计分析图

5.1 统计图

统计工具箱在 MATLAB 丰富的绘图功能上又添加了一些特殊的图形表现函数，以充分、直观地展现样本及其统计量的内在规律。例如，box 图用于描述数据样本，也用于通过图形来比较多个样本的均值。正态概率图是确定样本是否为正态分布的图形。分位数-分位数图用于比较两个样本的分布。下面展开介绍。

5.1.1 样本图

boxplot 函数

功能：数据样本的 box 图。其调用格式如下：

boxplot(X)：为 X 中的每列数据绘制一个 box 图。

boxplot(X,notch)：notch 取 1 时，绘制带切口的 box 图；默认 (notch=0) 时，box 图无切口。

boxplot(X, notch, 'sym')：'sym' 为野标记符号（如果有野值的话），默认符号为 “+”。

boxplot(X, notch, 'sym', vert, whis)：vert 控制 box 图水平放置还是垂直放置。当 vert=0 时，box 图水平放置；当 vert=1 时（默认），box 图垂直放置；whis 定义虚线的长度为内四分位间距 (IQR) 的函数（默认情况为 $15 \times \text{IQR}$ ）。如果 whis=0，则 box 图用 'sym' 规定的标记显示 “盒子” 外所有的数据。

【例 5-1】数据样本的 box 图示例。

```
>> clear all;  
x1 = normrnd(5,1,100,1);  
x2 = normrnd(6,1,100,1);  
boxplot([x1,x2],'notch','on')  
figure;  
boxplot([x1,x2],'notch','on','whisker',1)
```

运行程序，效果如图 5-1 及图 5-2 所示。

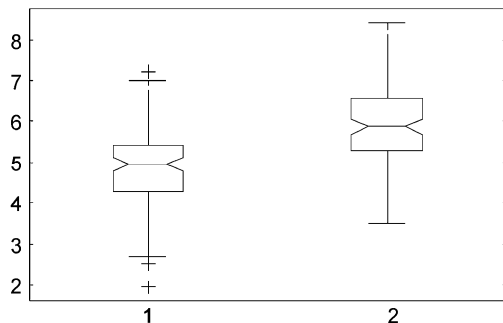


图 5-1 notch=0 时的 box 图

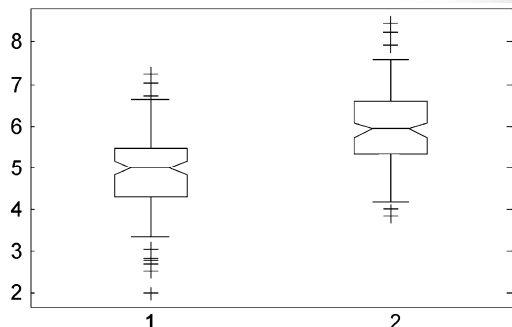


图 5-2 whisker=1 的 box 图

5.1.2 误差图

1. errorbar 函数

功能：误差条图。其调用格式如下：

errorbar(Y,E)：绘制区间为 $[Y-E, Y+E]$ 的误差条图。

errorbar(X,Y,E)：给出其误差条关于 Y 对称的 X-Y 图。

errorbar(X,Y,L,U)：给出 X-Y 图以及由 L 和 U 规定的误差界限的误差条。误差条是距离点 (X, Y) 上面的长度为 U (i)，下面的长度为 L (i) 的直线。X, Y, L, U 的长度必须相同。如果它们都是矩阵，则每列数据给出一条单独的线。

errorbar(...,LineStyle)：LineStyle 为一字符串，规定线的类型。

【例 5-2】误差条图示例。

```
>> clear all;
X = 0:pi/10:pi
Y = sin(X)
E = std(Y)*ones(size(X))
errorbar(X,Y,E);
```

运行程序，效果如图 5-3 所示。

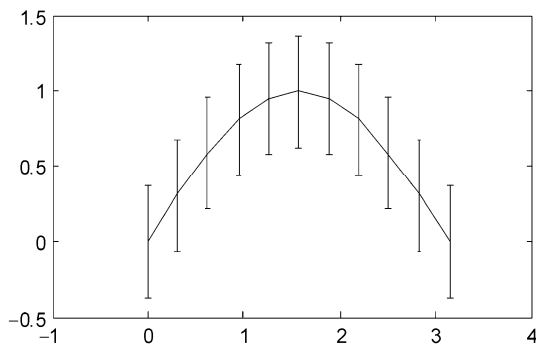


图 5-3 误差条图

2. rcoplot 函数

功能：回归残差图。其调用格式如下：

rcoplot(r,rint)：将试验样本回归后的残差及其置信区间绘制成误差条图，其中 r 和 rint 来自函数 regress 的输出。图中按数据的顺序给出各数据点的误差条。

【例 5-3】回归残差图示例。

```
>> clear all;
load moore
X = [ones(size(moore,1),1) moore(:,1:5)];
y = moore(:,6);
alpha = 0.05;
[betahat,lbeta,res,Ires,stats] = regress(y,X,alpha);
rcoplot(res,Ires)
title('残差图绘制');
ylabel('残差');xlabel('示例数据');
```

运行程序，效果如图 5-4 所示。

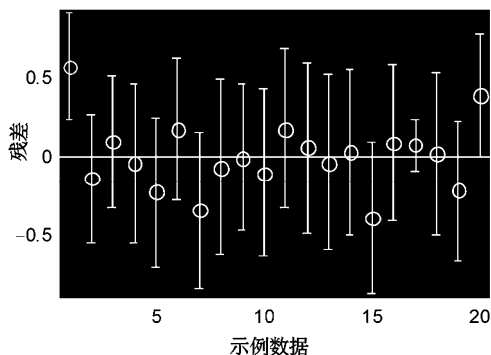


图 5-4 残差图的绘制效果

5.1.3 交互图

1. fsurfht 函数

功能：函数的交互轮廓图。其调用格式如下：

`fsurfht(fun,xlims,ylim)`

`fsurfht(fun,xlims,ylim,p1,p2,p3,p4,p5)`

参数说明：给出 `fun` 函数的交互轮廓图。其中 `fun` 为用户指定的函数名称。图中，`x` 轴的范围由 `xlims=[xmin, xmax]` 确定，`y` 轴的范围由 `ylim=[ylim, ymax]` 确定。另外把 5 个可选参数提供给函数 `fun`。函数 `fun` 的前两个参数分别是 `x` 变量和 `y` 变量。图中分别有一个垂直和水平的参考线（虚线），其交点为当前的 `x` 值和 `y` 值。同时既可观察到更新的 `x` 值（在图的顶部），也可在 `y` 轴和 `y` 轴相应的编辑框内输入一定的数而获得所计算的 `z` 值。

【例 5-4】绘制由 `gas.mat` 文件提供数据的高斯似然函数图形。编写一个 M 文件 `li5_4.m`。

```
function z = li5_4(mu,sigma,p1)
n = length(p1);
z = ones(size(mu));
for i = 1:n
z = z .* (normpdf(p1(i),mu,sigma));
end
```

`li5_4` 函数调用 `normpdf` 函数，其中视数据样本为固定量，而参数 μ 和 σ 为变量。假设汽油价格服从正态分布，现在来看看样本数据的似然函数曲面。

在命令窗口输入以下代码：

```
>> load gas
fsurfht('li5_4',[112 118],[3 5],price1)
```

运行程序，效果如图 5-5 所示。

其中，样本均值为图中 `x` 的最大值，但样本标准差并非图内 `y` 的最大值。

```
>> mumax=mean(price1)
mumax =
    115.1500
>> sigmamax=std(price1)*sqrt(19/20)
sigmamax =
     3.7719
```

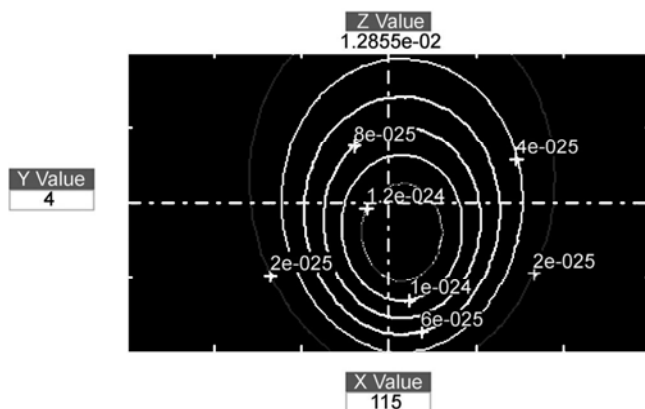



图 5-5 函数的交互轮廓图

2. gline 函数

功能：交互线绘制。其调用格式如下：

gline(h): 用鼠标在图 **h** 中点击两个点，可绘制出以此两点为端点的直线。点击第一点后，线的一端即固定，另一端为鼠标移动所在的位置，直至点击鼠标时，其所在位置即为线的另一端所在的位置。

gline: 在当前图中绘制线段。

hline = gline(...): 返回线段的句柄 **hline**。

【例 5-5】交互线绘制示例。

```
>> clear all;
x = 1:10;
y = x + randn(1,10);
scatter(x,y,25,'b','p')
lsline
mu = mean(y);
hold on
plot([1 10],[mu mu],'ro')
hline = gline;
set(hline,'Color','r')
```

运行程序，效果如图 5-6 所示。

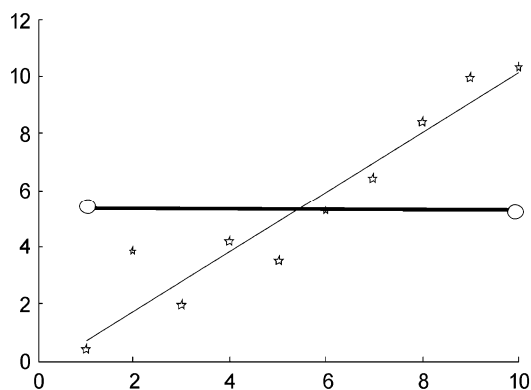


图 5-6 交互线的绘制

3. surfht 函数

功能：交互内插轮廓图。其调用格式如下：

surfht(Z)

surfht(x,y,Z)

参数说明：x 和 y 是曲线图中 x 轴和 y 轴的坐标矢量。x 的长度必须与 z 的列数相同，y 的长度必须与 z 的行数相同。根据给出的 x、y、z 数据，提供任意的 x、y 的坐标值上 z 的内插值。图中给出了水平和垂直参考线，其交点确定当前的 x 坐标和 y 坐标。可以任意拖拉参考线，或者在 x 轴和 y 轴相应的编辑框内输入坐标值，则同时能够直接观察到更新的 z 的内插值。

5.1.4 概率图

1. normplot 函数

功能：图形化正态性检验的正态概率图。其调用格式如下：

h = normplot(X)

参数说明：显示数据 x 的正态概率图。对于矩阵 x，为其每列显示一条线。图形以符号“+”标识样本数据。叠加在数据点上的实线是 x 的每列数据的第一和第三分位间的连线（以样本顺序统计量的鲁棒线性拟合）。此线延伸至样本的两端，有助于评价数据的线性程度。如果数据确实服从正态分布，则图形呈现为直线，而其他概率密度函数则表现出不同的弯曲。h 为返回曲线的句柄。

【例 5-6】产生一个正态分布样本绘制其他正态概率图。

```
>> clear all;
x = normrnd(10,1,25,1);
normplot(x)
title('正态概率图');
ylabel('概率'); xlabel('数据');
```

运行程序，效果如图 5-7 所示。

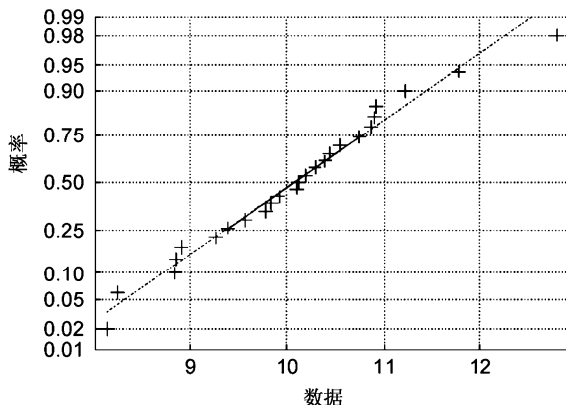


图 5-7 图形化正态性检验的正态概率图

2. pareto 函数

功能：帕累托图。其调用格式如下：

pareto(Y)

pareto(Y,names)

`H = pareto(...)`

参数说明：帕累托图亦称“主次因素排列图”，简称“排列图”。将矢量 `y` 中的每一个元素按元素数值递减顺序绘成直方条，并以其在 `y` 中的索引号进行标记。各直方条上方的折线显示累积频率。字符串矩阵 `name` 中的名称对 `y` 中相应的元素所绘的直方条进行标记。`h` 为返回图形的句柄。

【例 5-7】帕累托图示例。

```
>> clear all;
codelines = [200 120 555 608 1024 101 57 687];
coders = ...
{'Fred','Ginger','Norman','Max','Julia','Wally','Heidi','Pat'};
pareto(codelines, coders)
title('行代码')
```

运行程序，效果如图 5-8 所示。

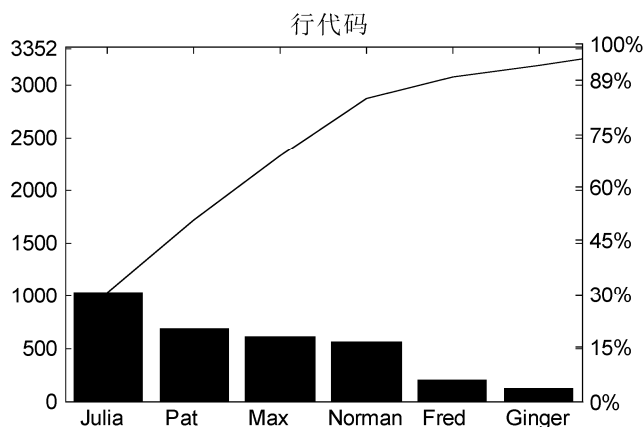


图 5-8 帕累托图

5.1.5 其他统计图

1. qqplot 函数

功能：两个样本的分位数-分位数图。其调用格式如下：

```
qqplot(X)
qqplot(X,Y)
qqplot(X,Y,pvec)
h = qqplot(X,Y,pvec)
```

参数说明：`qqplot(X,Y)`显示两个样本的分位数-分位数图。如果两个样本来源于不同的分布，那么，图中的曲线为直线。若 `X` 和 `Y` 为乱阵，则为它们的每列数据绘制单独的曲线。图中样本数据以“+”符号表示，并将位于第一分位数和第三分位数间的数据拟合绘制成一条线（这是两个样本顺序统计量的鲁棒线性拟合）。此线外推至样本数据的两端，以帮助用户评估数据的线性程度。

`qqplot(X,Y,pvec)`函数可在 `pvec` 矢量中规定分位数。

`h = qqplot(X,Y,pvec)`：返回线段的句柄。

【例 5-8】分位数图示例。

```
>> clear all;
x = poissrnd(10,50,1);
y = poissrnd(5,100,1);
qqplot(x,y);
xlabel('x 分位数');ylabel('y 分位数');
```

运行程序，效果如图 5-9 所示。

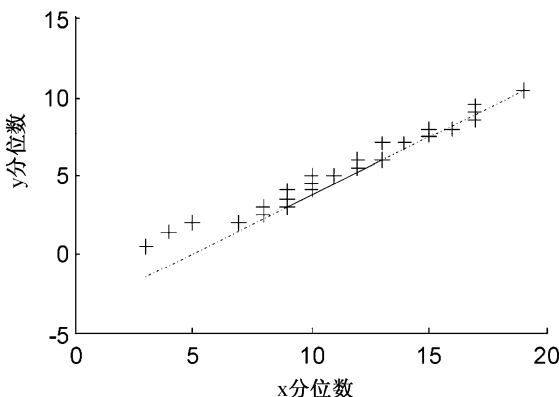


图 5-9 两个样本的分位数-分位数图

2. refcurve 函数

功能：在当前图中给出多项式拟合曲线。其调用格式如下：

refcurve(p)

refcurve

参数说明：在当前图中给出多项式 p （以系数矢量 p 表示）的曲线。 n 阶多项式函数为

$$y = p_1 x^n + p_2 x^{(n-1)} + \cdots + p_n x + p_{n+1}$$

hcurve = refcurve(...): 为返回曲线的句柄。

【例 5-9】使用 refcurve 函数合理增加人口数。

```
>> clear all;
p = [1 -2 -1 0];
t = 0:0.1:3;
y = polyval(p,t) + 0.5*randn(size(t));
plot(t,y,'ro')
h = refcurve(p);
set(h,'Color','r')
q = polyfit(t,y,3);
refcurve(q)
legend('数据','平均人口','合理增加','位置','NW')
```

运行程序，效果如图 5-10 所示。

3. reffline 函数

功能：在当前图中给出参考线。其调用格式如下：

reffline(m,b)

reffline(coeffs)

refline

hline = refline(...)

参数说明: `refline(m,b)`在图中给出斜率为 `m`、截距为 `b` 的直线。而 `refline(coeffs)`中, `coeffs` 为一个二元矢量, 所给出的直线如下:

$y = \text{coeffs}(2) + \text{coeffs}(1) * x$

`hline = refline(...)`返回直线的句柄。

`refline` 给出图中各线性对象的最小二乘拟合线 (以下线型除外: ‘-’, ‘--’, ‘.-’)。

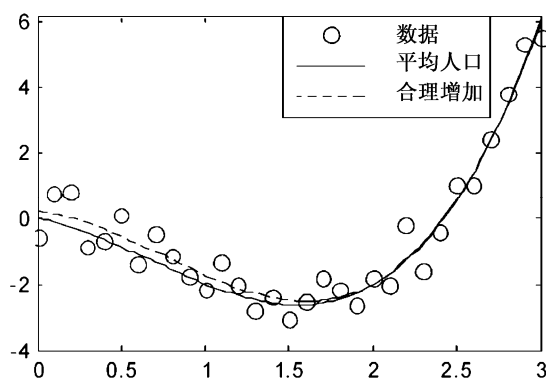


图 5-10 多项式拟合曲线

【例 5-10】在当前图中给出参考线示例。

```
>> clear all;
x = 1:10;
y = x + randn(1,10);
scatter(x,y,25,'b','*')
lsline
mu = mean(y);
hline = refline([0 mu]);
set(hline,'Color','r');
```

运行程序, 效果如图 5-11 所示。

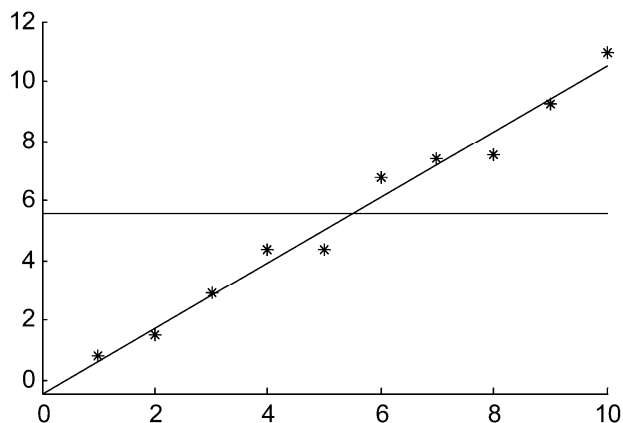


图 5-11 在当前图中给出参考线

5.2 统计工序管理图

统计工序管理（SPC）是指运用数理统计的一系列理论和方法，对生产过程或产品质量的管理和控制。各种抽样和验收方法，各种统计估计、比较和分析方法，产品质量和产生过程的管理图，以及其他各种统计图和分析图等都是统计质量管理常用的方法和工具，其核心是管理图。这种方法操作起来简单、易于在生产过程中实施。统计工具箱提供了主要的一些统计质量管理图函数。分别介绍如下：

5.2.1 工序图

1. capable 函数

功能：工序能力指数。其调用格式如下：

`p=capable(data, specs)`：计算样本数据落于给定上、下界的区间之外的概率。在此，假设数据矢量 `data` 中的测量值服从正态分布，其均值和方差均为常值，且测量是统计独立的。

`[p, Cp, Cpk]=capable(data, specs)`：则返回工序能力指数 `Cp` 和 `Cpk`，它们是工序满足质量要求程度的数值度量。

设在工序处于稳定状态下， T 为公差范围 $USL-LSL$ ， d 为质量特征的标准差， μ 为质量特征的分布中心， M 为公差中心。 C_p 是给定的公差范围 $USL-LSL$ 与六倍的标准差估计量之比，即

$$C_p = \frac{USL - LSL}{6\sigma}$$

对于一生产过程，当 μ 大致等于 M 时， $C_p=1$ 相当于每一千个产品中有稍多于一个的次品；而当每百万个产品中有一个次品时，其 C_p 值约为 1.6。 C_p 值越高，说明工序质量越高。当 M 与 C_p 不等时，则可用 C_{pk} 来描述，其表达式如下：

$$C_{pk} = \min\left(\frac{USL - \mu}{3\sigma}, \frac{\mu - LSL}{3\sigma}\right)$$

【例 5-11】假设规定一机器零件的尺寸误差在 3/1000 英寸内（误差服从正态分布）。如果在生产过程中，每个零件平均厚了 1/1000 英寸，同时其标准差为 1/1000 英寸，则工序能力指数是多少？

其实现的 MATLAB 程序代码如下：

```
>> clear all;
data=normrnd(1,1,30,1);
[p,Cp,Cpk]=capable(data,[-3,3]);
indices=[p Cp,Cpk]
```

运行程序，输出如下：

```
indices =
    0.1329    0.7692    0.3713
```

由结果可知，在每 1000 个零件中，大约有 17 个零件不符合规定要求。

2. capaplot 函数

功能：工序能力图。其调用格式如下：

`p = capaplot(data,specs)`

```
[p,h] = capaplot(data,specs)
```

参数说明：假设数据服从正态分布，其均值和方差未知， $p=\text{capaplot}(\text{data}, \text{specs})$ 将 data 向量中的观测值拟合成正态分布图，并返回新观测值落于 specs 规定的范围内的概率 p 在分布图中，由 specs 向量中的两个元素所界定的下界和上界之间的图形部分用阴影表示。 $[p,h] = \text{capaplot}(\text{data},\text{specs})$ 返回所绘图形的元素的句柄。

【例 5-12】假设规定一机器零件的尺寸误差在 3/1000 英寸内（误差服从正态分布）。如果在生产过程中，每个零件平均厚了 1/1000 英寸，同时其标准差为 1/1000 英寸。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
data = normrnd(3,0.005,100,1);
p=capaplot(data,[2.99 3.01])
grid on
title('工序能力指数')
```

运行程序，输出工序能力指数如下，效果如图 5-12 所示。

```
p =
    0.9608
```

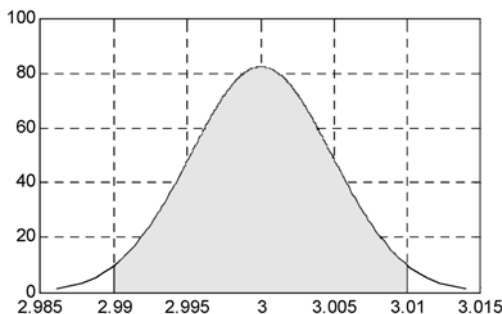


图 5-12 工序能力指数图

结果表明，新观测值落在 specs 规定的范围内的概率 P 为 96.08%。

5.2.2 密度图

1. histfit 函数

功能：附加正态密度曲线的直方图。其调用格式如下：

```
histfit(data)
histfit(data,nbins)
histfit(data,nbins,dist)
h = histfit(...)
```

参数说明：将 data 中的数据绘制成直方图。此时，图中矩阵的条数等于 data 中数据个数的平方根（取大于它的最小整数）。

【例 5-13】附加正态密度曲线的直方图示例。

```
>> clear all;
r = normrnd(10,1,100,1);
histfit(r)
h = get(gca,'Children');
```

```
set(h(2),'FaceColor',[.8 .8 1])
```

运行程序，效果如图 5-13 所示。

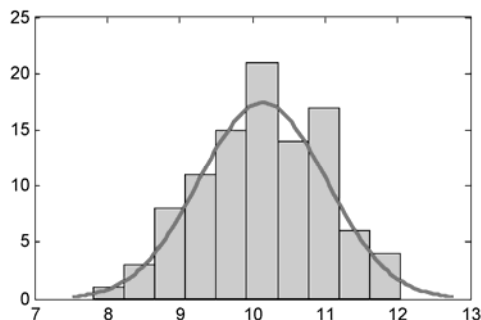


图 5-13 附加正态密度曲线的直方图

2. normspec 函数

功能：绘制规定区间的正态分布密度曲线。其调用格式如下：

```
normspec(specs)
```

```
normspec(specs,mu,sigma)
```

```
p = normspec(...)
```

```
[p,h] = normspec(...)
```

参数说明：绘制由矢量 `specs` 的两个元素所确定的上限和下限间的正态分布密度曲线。`mu` 和 `sigma` 为此正态分布的参数。除绘制曲线外，还返回样本落于其上限和下限间的概率。`h` 为曲线对象的句柄。如果 `specs` 矢量的第一个元素为 `-Inf`，即下限负无穷；同样，如果 `specs` 矢量的第二个元素为 `Inf`，即上限为正无穷。`h` 为返回密度曲线的句柄。

【例 5-14】绘制规定区间的正态分布密度曲线示例。

```
>> clear all;
p = normspec([1-3/128,Inf],1,2/128,'outside')
title('概率的区间');
xlabel('关键值'); ylabel('密度')
grid on;
```

运行程序，输出概率区间如下，效果如图 5-14 所示。

```
p =
    0.0668
```

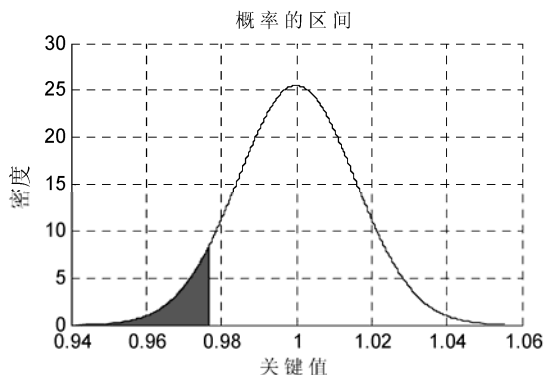


图 5-14 绘制规定区间的正态分布密度曲线

5.2.3 密度、平均、均值图

1. ewmaplot 函数

功能：指数加权滑动平均图（EWMA）。其调用格式如下：

`ewmaplot(data)`：绘制数据样本的 EWMA 图。`data` 中的行数据是按时间顺序给出的观测值。

`ewmaplot(data, lambda)`：由参数 `lambda` 的取值规定当前预测值受此前观测数据影响的程度，从而绘制数据样本的 EWMA 图。`lambda` 的取值区间为 $[0, 1]$ 。

`ewmaplot(data, lambda, alpha)`：规定了所绘制的置信区间的上下界的显著性水平。`alpha` 的默认值为 0.01，这意味着图中所绘制数据点中大约有 99% 落于此区间内。

`ewmaplot(data, lambda, alpha, specs)`：`specs` 矢量（包含两个元素）给出了所绘制图的相对响应结果规定的上下界线。

`h=ewmaplot(...)`：返回各曲线的句柄矢量。

【例 5-15】考虑一个均值随时间缓慢漂移的工序。监控这种工序，采用 EWMA 图比均值管理图更可取。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
t=(1:28)';
r=normrnd(10+0.02*t(:,ones(4,1))),0.5);
ewmaplot(r,0.4,0.01,[9.75,10.75]);
title('指数加权移动平均图');
ylabel('加权移动平均'); xlabel('样本')
```

运行程序，效果如图 5-15 所示。

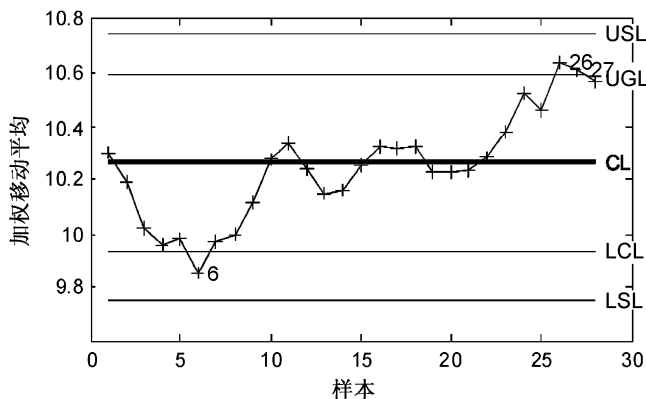


图 5-15 指数加权移动平均图

2. schart 函数

功能：标准差管理图（s-图）。其调用格式如下：

`schart (data)`：绘制 `data` 中数据的标准差管理图，`data` 中的每行数据为按时间顺序给出的观测值，上下管理限 `UCL` 和 `LCL` 为此工序中新观测值的 99% 置信区间。

`schart (data, conf)`：可用 `conf` 给出自定置信区间管理限。如 `conf=0.95` 时，图形则绘出 95% 置信区间。

`schart(data, conf, specs)`：可用两个元素矢量 `specs` 限制所绘曲线的界限。

[outliers, h]=schart(data, conf, specs): 给出 data 中那些数据均值失控的行的序号 (返回至矢量 outliers 中), 另外还给出曲线的句柄 (矢量 h)。

【例 5-16】给出新生产的零件尺寸的标准差管理图。

生产过程中每小时测一次, 共监测了 36 h。矩阵 runout 中每一行包含 4 个随机抽取零件的尺寸。

```
>> clear all;
load parts
schart(runout)
title('s-图');
xlabel('样本数据'); ylabel('标准差');
运行程序, 效果如图 5-16 所示。
```

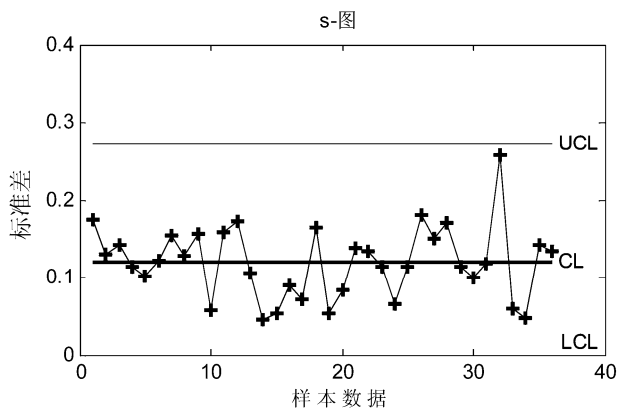


图 5-16 标准差管理图

3. xbarplot 函数

功能: 均值管理图。其调用格式如下:

xbarplot(data): 给出 data 数据的均值管理图。data 中的每行数据为按时间顺序给出的观测值, 上下管理限 UCL 和 LCL 为此工序中新观测值的 99%置信区间。

xbarplot(data, conf): 可用 conf 给出自定的置信区间管理限。如果 conf=95%时, 图形则绘出 95%置信区间。

xbarplot(data, conf, specs, 'sigmaest'): 可用两个元素矢量 specs 限制所绘曲线的界限。

[outlier, h]= xbarplot(...): 给出 data 中那些数据均值失控的行的序号 (返回至矢量 outliers) 中。另外还给出曲线的句柄 (矢量 h)。

【例 5-17】绘制均值管理图。

```
>> clear all;
load parts
schart(runout,0.999,[-0.5 0.5]);
title('xbar-图');
xlabel('样本'); ylabel('均值');
```

运行程序, 效果如图 5-17 所示。

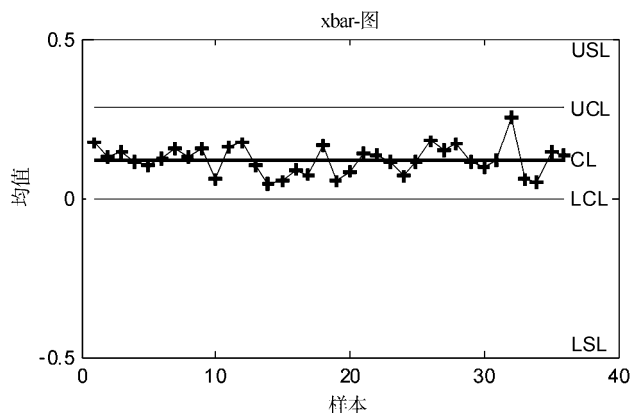


图 5-17 均值管理图

5.3 频率分布表与频率直方图

频率分布表是一种对连续变量的观测数据进行分组整理和初步分析的一种重要的统计数据表。频率直方图是频率分布表的图形化。通过频率分布表与频率直方图，可以实现对变量分布形态（概率密度曲线）的初步估计。掌握频率分布表的编制与频率直方图的绘制方法是统计应用的一项基本技能。

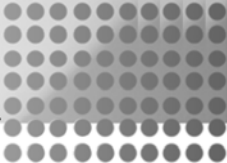
下面举例说明频率分布表的编制方法和频率直方图的绘制。

【例 5-18】钢材中的含硅量 X 是影响材料性能的一项重要因素。在炼钢生产过程中，由于各种随机因素的影响，各炉钢的含硅量 X 是有差异的。对含硅量 X 的概率分布的了解是有关钢材料性能分析的重要依据。某炼钢厂 120 炉正常生产的 25MnSi 钢的含硅量（单位：%）如下：

0.86 0.83 0.77 0.81 0.81 0.80 0.79 0.82 0.82 0.81
 0.82 0.78 0.80 0.81 0.87 0.81 0.77 0.78 0.77 0.78
 0.77 0.71 0.95 0.78 0.81 0.79 0.80 0.77 0.76 0.82
 0.84 0.79 0.90 0.82 0.79 0.82 0.79 0.86 0.81 0.78
 0.82 0.78 0.73 0.84 0.81 0.81 0.83 0.89 0.78 0.86
 0.78 0.84 0.84 0.75 0.81 0.81 0.74 0.78 0.76 0.80
 0.75 0.79 0.85 0.78 0.74 0.71 0.88 0.82 0.76 0.85
 0.81 0.79 0.77 0.81 0.81 0.87 0.83 0.65 0.64 0.78
 0.80 0.80 0.77 0.84 0.75 0.83 0.90 0.80 0.85 0.81
 0.82 0.84 0.85 0.84 0.82 0.85 0.84 0.82 0.85 0.84
 0.81 0.77 0.82 0.83 0.82 0.74 0.73 0.75 0.77 0.78
 0.87 0.77 0.80 0.75 0.82 0.78 0.78 0.82 0.78 0.78

保存上面数据为 li5_17data.mat 文件，下面介绍如何编制频率分布表，以及绘制频率直方图的 MATLAB 实现方法。

```
>> clear all;  
load li5_17data.mat
```



1. 数据分组

- (1) 确定数据组个数。根据样本容量 n 确定分组数 k ，推荐公式为 $k = 1.87(n - 1)^{2/5}$ 。
- (2) 计算极差。计算公式为 $R_n = \max(x_1, x_2, \dots, x_n) - \min(x_1, x_2, \dots, x_n)$ 。
- (3) 确定组距。计算公式为 $d \approx R_n/k$ ，一般取 d 为数据的最小测量单位的整数倍。
- (4) 确定各组端点。计算公式为 $a_k = a_0 + dk$ ($k = 0, 1, \dots, n$)，其中， $a_0 < \min\{x\}$ ， $a_n > \max\{x\}$ 。

a_0 的确定方法：一般地取 a_0 比数据的最小值小半个测量单位。

2. 统计各组频数

各组频数就是数据落入各个小组中的个数，记为 n_i 。

上述计算的 MATLAB 实现由两步完成：第一步，先确定数组的推荐公式，求出分组数 k ；第二步，其他的计算极差、确定组距、确定各组端点和统计各组频数的工作均可由 MATLAB 系统函数 hist 完成。hist 的输入参数通常有两个：第一个是数据向量，第二个是小组个数。hist 的输出参数有两个：第一个返回各组的数据频数，第二个返回各个数据组的区间位置值（组中值）。

```
k=ceil(1.87*(length(X)-1)^0.4);
[ni,ak]=hist(X,k);
```

3. 计算频率

- (1) 计算各组频率。计算公式为 $f_i = n_i/n$ 。MATLAB 代码如下：

```
>> fi=ni/length(X);
```

- ② 计算各组累积频率。计算公式为 $F_i = \sum_{j=1}^i f_j$ ($i = 1, 2, \dots, k$)。MATLAB 计算指令为：

```
>> mfi=cumsum(fi);
```

4. 编制频率分布表

逐一运行上述 MATLAB 计算指令，再运行指令

```
>> stats=[1:k]',ak',ni',fi',mfi']
```

就可得到 120 炉 25MnSi 钢的含硅量数据的频率分布表，稍加整理即得到表 5-1。

表 5-1 120 炉的 25MnSi 钢的含硅量数据频率分布表

组序	组中值	频数	频率	累积频率
1.0000	0.6519	2.0000	0.0167	0.0167
2.0000	0.6758	0	0	0.0167
3.0000	0.6996	2.0000	0.0167	0.0333
4.0000	0.7235	2.0000	0.0167	0.0500
5.0000	0.7473	8.0000	0.0667	0.1167
6.0000	0.7712	29.0000	0.2417	0.3583
7.0000	0.7950	15.0000	0.1250	0.4833
8.0000	0.8188	36.0000	0.3000	0.7833
9.0000	0.8427	15.0000	0.1250	0.9083
10.0000	0.8665	6.0000	0.0500	0.9583
11.0000	0.8904	4.0000	0.0333	0.9917
12.0000	0.9142	0	0	0.9917
13.0000	0.9381	1.0000	0.0083	1.0000



接下来介绍频率直方图和累积频率折线图及其绘制方法。

频率直方图是连续性变量频率分布的图形化，累积频率折线图是累积频率分布的图形化。

在频率直方图中，横轴表示观测变量的观测者，每一个小矩形的水平边长=组距；纵轴表示各组数据的频率，由于频率密度曲线下方的面积恒等于 1，因此为保证直方图中所有矩形条面积之和也等于 1，规定每个小矩形的高度=该组数据的频率/组距。

用 MATLAB 绘制直方图的指令是 `hist` 或 `histfit`，但是需要指出的是，为了观察上的方便，这两个指令绘制出的图形纵轴的刻度是频数值。

`hist` 前面已经见过，当有输出参数时，它将完成各组频数的统计工作；若无输出参数，则直接绘制频率直方图。

```
>> hist(X) %画直方图
h=findobj(gca,'Type','patch'); %为修饰图形提取指定属性对象的图形句柄 h
set(h,'FaceColor','y','EdgeColor','b'); %修饰,设置直方图的线条颜色与填充色
```

运行程序，效果如图 5-18 所示。

`histfit` 指令在绘制频率直方图的同时附加一条正态密度曲线。

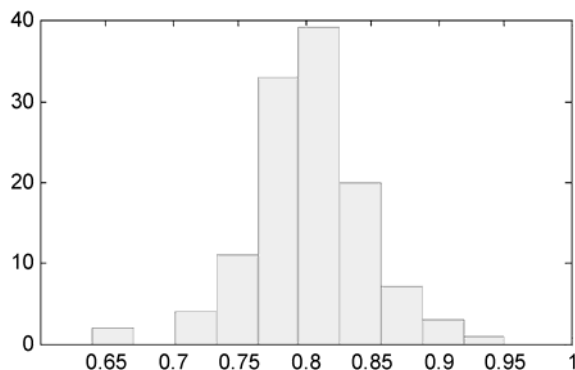


图 5-18 `hist` 指令绘制的直方图

```
h=histfit(X, 13); %画附正态参考曲线的直方图，并提取图形句柄 h
set(h(1),'FaceColor','c','EdgeColor','w'); %修饰,设置直方图线条颜色与填充色
set(h(2),'Color','r'); %修饰,设置正态参考曲线的颜色
```

运行指令，效果如图 5-19 所示。

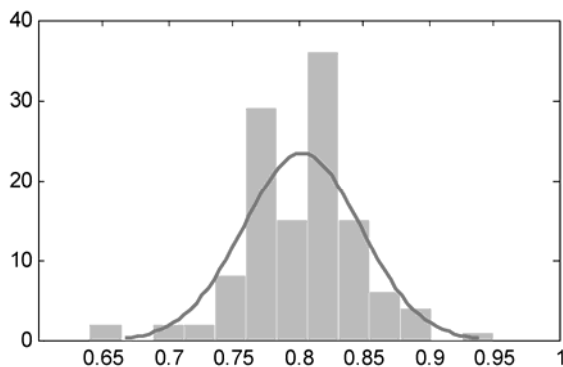


图 5-19 `histfit` 指令绘制的直方图

有时，人们常以频率分布表中的组中值为横坐标、以累积频率为纵坐标绘制累积频率折线（请读者用 `plot` 指令自行画出图形）。

在应用中，可以根据频率直方图（累积频率折线图）了解变量的概率密度曲线（分布曲线）的大致形态，进而估计变量的分布类型。在得出初步的结论后应继续通过分布参数的估计和分布拟合检验得出更为精细的结论。

对于离散型随机变量，一般在大量条件下求样本数据的频率，画出不同数据点频率值的火柴杆图（或散点图），通过对已知的离散分布的分布律图形作出变量分布形态的估计，进一步分析参考，这里不再作介绍。

5.4 非线性回归模型

非线性回归模型（Nonlinear Regression Models）是其回归函数关于未知参数具有非线性结构的回归模型。模型的拟合一般很困难，通常首先需猜测未知参数的初始值，然后反复迭代。每次迭代都会修正当前的估测值，直至算法收敛为止。

拟合以下形式的非线性模型：

$$y = f(\mathbf{X}, \boldsymbol{\beta}) + \varepsilon$$

式中 y —— $n \times 1$ 观测得到；

f —— X 和 β 的任何一个函数；

\mathbf{X} —— $n \times p$ 输入向量矩阵；

$\boldsymbol{\beta}$ —— $p \times 1$ 待估计的未知参数向量；

ε —— $n \times 1$ 随机扰动向量。

5.4.1 非线性拟合

1. `nlinfit` 函数

功能：用高斯-牛顿方法进行非线性最小二乘拟合。其调用格式如下：

```
beta = nlinfit(X,y,fun,beta0)
```

```
[beta,r,J,COVB,mse] = nlinfit(X,y,fun,beta0)
```

```
[...] = nlinfit(X,y,fun,beta0,options)
```

返回由 ‘model’ 描述的线性函数的系数。‘model’ 是用户提供的函数，其形式如下：

$$\hat{y} = f(\boldsymbol{\beta}, \mathbf{X})$$

即在给定独立变量 \mathbf{X} 和参数估计量 $\boldsymbol{\beta}$ 后，‘model’ 返回的预测值。矩阵 \mathbf{X} 中，每个独立变量有一列数据。响应结果 \mathbf{y} 为一列矢量，其元素个数等于 \mathbf{X} 的行数。

【例 5-19】用高斯-牛顿方法进行非线性最小二乘拟合示例。

```
>> load reaction %MATLAB 自带的数据库
```

```
[beta,r,J] = nlinfit(reactants,rate,@hougen,beta)
```

运行程序，输出如下：

```
beta = %拟合系数
    1.2526
    0.0628
    0.0400
    0.1124
```

```

1.1914
r = %残差
0.1321
-0.1642
-0.0909
0.0310
0.1142
0.0498
-0.0262
0.3115
-0.0292
0.1096
0.0716
-0.1501
-0.3026
J = %雅可比矩阵
6.8739 -90.6525 -57.8634 -1.9288 0.1614
3.4454 -48.5350 -13.6239 -1.7030 0.3034
5.3563 -41.2094 -26.3039 -10.5216 1.5095
1.6950 0.1091 0.0186 0.0278 1.7913
2.2967 -35.5653 -6.0537 -0.7567 0.2023
11.8669 -89.5648 -170.1730 -8.9565 0.4400
4.4973 -14.4261 -11.5409 -9.3769 2.5744
4.1831 -41.7891 -16.8935 -5.7794 1.0082
11.8285 -51.3718 -154.1151 -27.7408 1.5001
9.1514 -25.5946 -76.7838 -30.7135 2.5790
3.3373 0.0900 0.0720 0.1079 3.5269
9.3663 -102.0600 -107.4317 -3.5811 0.2200
4.7512 -24.4628 -16.3086 -10.3001 2.1141
    
```

2. nlintool 函数

功能：拟合非线性方程并绘制交互式图形。其调用格式如下：

nlintool(X,y,fun,beta0): 提供对数据 (x, y) 进行非线性拟合的预测图形。其中，用两条红色曲线给出 95% 全局置信区间。**beta0** 是包含参数的初始估计测值的矢量。

nlintool(X,y,fun,beta0,alpha): 区别在于其给出预测值的 100 (1-alpha) % 置信区间。**nlintool** 显示图形的“矢量”。响应变量 y 是与 x 的行数相匹配的列矢量。**alpha** 的默认值为 0.05，即对应 95% 的置信区间。

nlintool(X,y,fun,beta0,alpha,'xname','yname'): 可以在图中的 X 轴和 Y 轴分别标上自定的‘xname’和‘yname’名称。拖拉白色参考线（虚线）可同时观察更新的预测值；或在参数 x 的编辑框内输入参数值，同样可以获得相应的预测值。使用‘Model’的菜单可交互地改变模型；使用“Export”菜单可将特定的变量移至基本工作台。

【例 5-20】拟合非线性方程并绘制交互式图形示例。

```

>> load reaction
nlintool(reactants,rate,@hougen,beta,0.01,xn,yn)
    
```

运行程序，效果如图 5-20 所示。

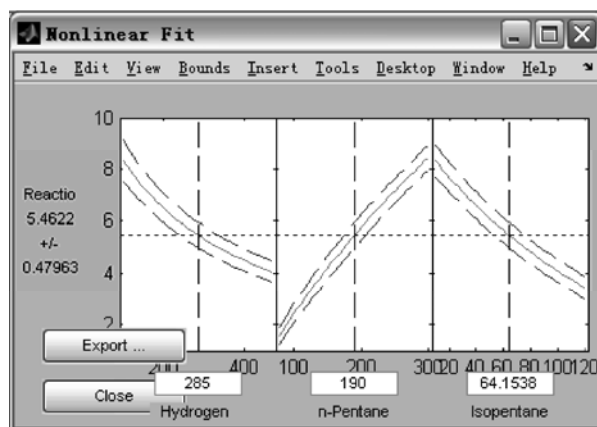


图 5-20 拟合非线性交互式图形界面

5.4.2 置信区间

1. nlparci 函数

功能：非线性模型的参数估计的置信区间。其调用格式如下：

```
ci = nlparci(beta,resid,'covar',sigma)
```

```
ci = nlparci(beta,resid,'jacobian',J)
```

```
ci = nlparci(...,'alpha',alpha)
```

【例 5-21】非线性模型的参数估计的置信区间示例。

```
>> load reaction
[beta,resid,J,Sigma] = ...
    nlinfit(reactants,rate,@hougen,beta);
ci = nlparci(beta,resid,'jacobian',J)
ci =
    -0.7467    3.2519
    -0.0377    0.1632
    -0.0312    0.1113
    -0.0609    0.2857
    -0.7381    3.1208
```

2. nlpredci 函数

功能：非线性模型预测置信区间。其调用格式如下：

```
[ypred,delta] = nlpredci(modelfun,x,beta,resid,'covar',sigma)
```

```
[ypred,delta] = nlpredci(modelfun,x,beta,resid,'jacobian',J)
```

```
[...] = nlpredci(...,param1,val1,param2,val2,...)
```

【例 5-22】非线性模型预测置信区间示例。

```
>> load reaction;
[beta,resid,J,Sigma] = nlinfit(reactants,rate,@hougen,...
    beta);
newX = reactants(1:2,:);
[ypred,delta] = nlpredci(@hougen,newX,beta,resid,...
    'Covar',Sigma)
ypred =
```



```
8.4179
3.9542
delta =
0.2805
0.2474
```

5.5 主成分分析

多元分析，是基于同时对多个对象或变量联合观测所得的多元数据的分析，研究多个变量或多个属性表征的多元总体。

多元分析的主要方法，有主成分分析、聚类分析、典型相关分析，以及回归分析、方差分析和协方差分析等。而主成分分析可以说是最常用的一种方法，统计工具箱所提供的多元统计方法即为主成分分析。

主成分分析亦称“主分量分析”或“分量分析”等，是将多个相关变量简化为少数几个不相关变量的一种多元统计方法。其目的在于简化统计数据和揭示变量间的关系。每个主成分是初始变量的线性组合，所有的主成分间相互正交，所以没有冗余信息，它们构成数据空间的正交集。从数学的角度看，其根本思想在于降维，而降维是从简化方差和协方差的结构来考虑的。

主成分的全体决定于变量正交集的维数。但通常情况下，前几个主成分的方差之和会超过初始数据方差总和的 80%。

多元统计的固有困难之一是多维变量的可视化问题。在 MATLAB 中，plot 函数能够显示二维变量关系图形；plot3 和 surf 函数能够显示不同的三维视图；当变量超过 3 个时，则需要通过想象把握它们之间的关系。

5.5.1 巴特力特检验

barttest 函数

功能：主成分的巴特力特检验。其调用格式如下：

ndim = barttest(X,alpha)：在显著性水平基础上，返回非随机矩阵 **X** 的必要维数。它由一系列假设检验所确定。**ndim=1** 表明数据 **X** 对应于每个主成分的方差是相同的；**ndim=2** 表明数据 **X** 对应于第二成分及其余成分的方差是相同的；依次类推。

[ndim,prob,chisquare] = barttest(X,alpha)：则返回模型维数、假设检验的显著性值和检验的 X^2 值。

巴特力特检验是一种等方差性检验。

【例 5-23】主成分的巴特力特检验示例。

```
>> clear all;
X = mvnrnd([0 0],[1 0.99; 0.99 1],20);
X(:,3:4) = mvnrnd([0 0],[1 0.99; 0.99 1],20);
X(:,5:6) = mvnrnd([0 0],[1 0.99; 0.99 1],20);
[ndim, prob] = barttest(X,0.05)
ndim =
    3
prob =
    0
```

```

0
0
0.6749
0.6609

```

5.5.2 PCA

1. pcacov 函数

功能：运用协方差矩阵进行主成分分析。其调用格式如下：

```
COEFF = pcacov(V)
```

```
[COEFF,latent] = pcacov(V)
```

```
[COEFF,latent,explained] = pcacov(V)
```

说明：参数协方差矩阵 V 进行主成分分析，返回主成分 $COEFF$ 、协方差矩阵 X 的特征值 ($latent$) 和每个特征向量表征在观测量总方差中所占的百分数 ($explained$)。

【例 5-24】运用协方差矩阵进行主成分分析示例。

```

>> load hald
covx = cov(ingredients);
[COEFF,latent,explained] = pcacov(covx)
COEFF =
    -0.0678    0.6460   -0.5673    0.5062
    -0.6785    0.0200    0.5440    0.4933
     0.0290   -0.7553   -0.4036    0.5156
     0.7309    0.1085    0.4684    0.4844
latent =
    517.7969
     67.4964
     12.4054
      0.2372
explained =
     86.5974
     11.2882
      2.0747
      0.0397

```

2. pcares 函数

功能：主成分分析的残差。其调用格式如下：

```
residuals = pcares(X,ndim)
```

```
[residuals,reconstructed] = pcares(X,ndim)
```

说明：返回保留 X 的 $ndim$ 个主成分所获得的残差。注意， $ndim$ 是一个标量，必须小于 X 的列数。而且， X 是数据矩阵，而不是协方差矩阵。

【例 5-25】主成分分析的残差示例。

```

>> load hald
r1 = pcares(ingredients,1);
r2 = pcares(ingredients,2);
r3 = pcares(ingredients,3);
>> r11 = r1(1,:);

```

```
r11 =
    2.0350    2.8304   -6.8378    3.0879
>> r21 = r2(1,:)
r21 =
   -2.4037    2.6930   -1.6482    2.3425
>> r31 = r3(1,:)
r31 =
    0.2008    0.1957    0.2045    0.1921
```

3. princomp 函数

功能：主成分分析。其调用格式如下：

```
[COEFF,SCORE] = princomp(X)
```

```
[COEFF,SCORE,latent] = princomp(X)
```

```
[COEFF,SCORE,latent,tsquare] = princomp(X)
```

```
[...] = princomp(X,'econ')
```

说明：对数据矩阵 X 进行主成分分析，给出各主成分（PC）、所谓的 Z-得分（SCORE）、 X 的方差矩阵的特征值（latent）和每个数据点的 Hotelling T2 统计量（tsquare）。

【例 5-26】监测湖泊水质，设 15 个监测点，每个监测点监测指标为 5 项（表 5-2），用主成分分析法确定它最佳的布设点。

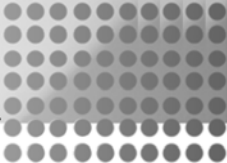
表 5-2 水质监测数据表

(mg/L)

点位	DO	COD	BOD	T-N	T-P
1	4.3	4.74	4.23	3.66	0.105
2	5.9	4.61	2.59	2.92	0.081
3	7.0	3.94	2.92	1.71	0.072
4	6.9	3.92	3.11	1.32	0.075
5	7.4	4.02	3.10	1.26	0.076
6	6.9	3.75	3.15	1.05	0.096
7	6.7	4.44	3.14	1.02	0.072
8	6.8	4.35	4.08	1.27	0.110
9	6.2	4.24	2.33	0.71	0.068
10	7.4	3.99	2.84	0.74	0.063
11	8.1	4.43	3.44	0.86	0.070
12	7.7	4.31	3.50	0.93	0.074
13	5.7	4.88	5.02	1.84	0.134
14	6.8	4.73	4.34	1.39	0.109
15	5.5	5.93	5.06	2.81	0.240

其实现的 MATLAB 程序代码如下：

```
>> clear all;
load li5_26data.mat
stdr=std(x);
sr=x./stdr(ones(15,1),:);
```



```
[pcs,newdata,variances,t2]=princomp(sr);
variances'           %特征值
(100*variances/sum(variances))'           %特征值贡献率
pcs(:,1:2)'
```

运行程序，输出如下：

```
variances' =
    3.5195    0.9347    0.2503    0.1686    0.1268
(100*variances/sum(variances))' =
    70.3898    18.6946    5.0061    3.3725    2.5370
pcs(:,1:2)' =
    0.4180   -0.4836   -0.4336   -0.4230   -0.4736
    0.5645    0.2255    0.4508   -0.5528    0.3489
```

pcs 的值分别代表五项指标在主成分中的权系数，即作用大小。从污染角度出发，根据各指标在主成分中的作用大小，分别给第 I、II 主成分赋予物理意义。从 pcs 的值可看出，在第 I 主成分中，第 1 项（DO）对水质的影响是正的，而第 2~5 项（分别为 COD、BOD、T-N、T-P）对水质的影响是负的，主要反映了有机污染物和水质自净作用的对比程度，这些值越大，说明水质质量越好，自净能力强；而第 II 主成分主要反映环境单元在第 I 主成分值大体固定的条件下水体中的氨的形成富营养化程度的量度，随着 T-N 项权值的增加，说明富营养化引起水质质量的下降。

```
>> newdata(:,1:2);           %主成分的得分
plot(newdata(:,1),newdata(:,2),'rp'); %主成分效果图
```

运行程序，效果如图 5-21 所示。

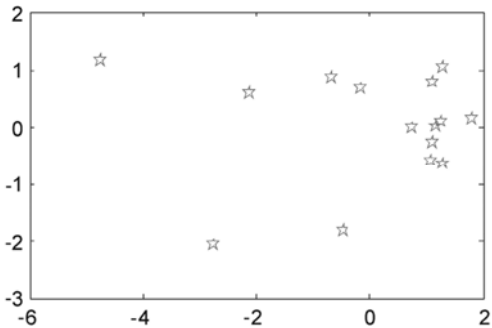


图 5-21 主成分得分图

从图中可以看出，15 个观察点被分成六类：（3，4，5，6，9，10，11，12），（8，14），（1），（2），（13），（15）。而这六类在二维平面图上是按照一定的方向和顺序依次排列的，自右到左，污染程度逐渐增加。不同的污染类别，实质上是客观反映了沿岸工业、人口分布对水环境的影响，两相邻类在污染类型上具有一定的相似性，而在污染程度上具有显著的差异性。从分类结果看，尚需进一步优选的类别有（3，4，5，6，9，10，11，12），（8，14）两类，可根据中点位的主成分值相差最大的原则选择，参考得分值，明显最佳点为（10），（14）。至此，15 个观察点经优选后的最佳点位为（1，2，10，13，14，15）。

在应用主成分分析时，主成分数目的估计是一个应该注意的问题。在计算的主成分中，第一个主成分最重要，随着主成分的增加，重要程度依次降低，以至到后来的许多主成分反映的是噪声和误差信息。进行数据分析时，引入的主成分过少，则不能说明要解决的问题，而引入



过多，则引入噪声及误差。因此合理确定主成分数目是充分利用化学量测数据信息和滤除噪声的有效方法之一。

5.6 实验设计

数据与其中所含的信息是存在差别的。若要从数据中提取信息，则需对产生数据的系统作出一定的假设。运用这些假设和实际理论可以建立系统的数学模型。

通常，即便严格用公式表达的模型仍有未知的常量。实验的目的就是获得数据以估计这些常数。在系统运转过程中可以进行观测研究，这样迟早会获得需用的所有数据。事实上，这是相当通用的方法。而在统计建模中所关心的历史数据一般有如下三个特征。

(1) 观测系统中各变量的变化，以及系统输出随之相应的变化情况。需要注意的是，系统的变化并不一定意味着其输出的变化。

(2) 统计建模中通常假设各观测量是相互独立的。但系统运转过程中，实际情况并不一定如此。

(3) 控制具体系统的运转常常是依次改变控制变量来观测结果。倘若同时改变两个以上变量，则各变量对结果的影响是无法隔离的。

实验设计可以直接解决这些问题，其最大的优点是能够主动地操纵所研究的系统。它比被动地进行测量需要的数据量少，但获取信息的质量却较高。

下面针对不同的情况进行介绍。

5.6.1 优化设计

1. cordexch 函数

功能：使用坐标交换进行 D-优化设计。其调用格式如下：

```
dCE = cordexch(nfactors,nruns)
```

```
[dCE,X] = cordexch(nfactors,nruns)
```

```
[dCE,X] = cordexch(nfactors,nruns,model)
```

```
[dCE,X] = cordexch(...,param1,val1,param2,val2,...)
```

说明：在初始设计的实验（startdes）基础上再给出 nruns 次新的实验因子设置。‘model’ 的意义同前所述，用来控制模型的阶数，可选项为 ‘interaction’、‘quadratic’ 和 ‘purequadratic’。paramN 和 valN 为其有效参数及其参数值的设置。

【例 5-27】使用坐标交换进行 D-优化设计示例。

```
>> nfactors = 3;
nruns = 7;
[dCE,X] = cordexch(nfactors,nruns,'interaction','tries',10)
dCE =
    1     1    -1
    1    -1    -1
   -1     1     1
   -1    -1     1
   -1    -1    -1
   -1    -1    -1
    1     1     1
```

```

      1    -1    1
X =
      1      1      1     -1      1     -1     -1
      1      1     -1     -1     -1     -1      1
      1     -1      1      1     -1     -1      1
      1     -1     -1      1      1     -1     -1
      1     -1     -1     -1      1      1      1
      1      1      1      1      1      1      1
      1      1     -1      1     -1      1     -1

```

2. dcovary 函数

功能：规定了固定方差的 D-优化设计。其调用格式如下：

```
dCV = dcovary(nfactors,fixed)
```

```
[dCV,X] = dcovary(nfactors,fixed)
```

```
[dCV,X] = dcovary(nfactors,fixed,model)
```

```
[dCV,X] = daugment(...,param1,val1,param2,val2,...)
```

说明：给出满足于固定方差的 fixed 的 D-优化设计，nfactors 为所希望陈旧变元数。‘model’ 的意义和选项与前述相同。paramN 和 valN 为其有效参数及其参数值的设置。

【例 5-28】规定了固定方差的 D-优化设计示例。

```

>> time = linspace(-1,1,8)';
[dCV1,X] = dcovary(3,time,'linear')
dCV1 =
    1.0000    1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -0.7143
    1.0000   -1.0000    1.0000   -0.4286
   -1.0000    1.0000    1.0000   -0.1429
   -1.0000   -1.0000   -1.0000    0.1429
   -1.0000    1.0000    1.0000    0.4286
    1.0000   -1.0000    1.0000    0.7143
    1.0000    1.0000   -1.0000    1.0000
X =
    1.0000    1.0000    1.0000   -1.0000   -1.0000
    1.0000   -1.0000   -1.0000   -1.0000   -0.7143
    1.0000    1.0000   -1.0000    1.0000   -0.4286
    1.0000   -1.0000    1.0000    1.0000   -0.1429
    1.0000   -1.0000   -1.0000   -1.0000    0.1429
    1.0000   -1.0000    1.0000    1.0000    0.4286
    1.0000    1.0000   -1.0000    1.0000    0.7143
    1.0000    1.0000    1.0000   -1.0000    1.0000

```

3. rowexch 函数

功能：使用行交换进行 D-优化设计。其调用格式如下：

```
dRE = rowexch(nfactors,nruns)
```

```
[dRE,X] = rowexch(nfactors,nruns)
```

```
[dRE,X] = rowexch(nfactors,nruns,model)
```

```
[dRE,X] = rowexch(...,param1,val1,param2,val2,...)
```

参数说明：产生因子实验设置矩阵 dRE，其列数为 nfactors，行数为 nruns。对于 D-优化设

计，使用附加常数项的线性模型。并产生关联设置矩阵 X 。‘model’ 为给出拟合特定回归模型的设计。其输入参量为以下字符串之一：

- (1) ‘interaction’ ——包括常数项、线性项和交叉乘积项。
- (2) ‘quadratic’ ——任意基础上增添平方项。
- (3) ‘purequadratic’ ——包括常数项、线性项和平方项。

【例 5-29】使用行交换进行 D-优化设计示例。

```
>> nfactors = 3;
nruns = 7;
[dRE,X] = rowexch(nfactors,nruns,'interaction','tries',10)
dRE =
    -1    -1    -1
    -1    -1     1
     1     1     1
    -1     1    -1
    -1     1     1
     1    -1    -1
     1     1    -1
X =
     1    -1     1    -1     1     1     1
     1    -1    -1     1     1    -1    -1
     1     1     1     1     1     1     1
     1    -1     1    -1    -1     1    -1
     1    -1     1     1    -1    -1     1
     1     1    -1    -1    -1    -1     1
     1     1     1    -1     1    -1    -1
```

5.6.2 因子设计

1. ff2n 函数

功能：两种水平全因子设计。其调用格式如下：

$dFF2 = \text{ff2n}(n)$

说明：产生两种水平的设计实验矩阵 X 。 n 为因子数量， X 的列数为 n ，行数为 $2n$ 。

【例 5-30】两种水平全因子设计示例。

```
>> dFF2 = ff2n(3)
dFF2 =
     0     0     0
     0     0     1
     0     1     0
     0     1     1
     1     0     0
     1     0     1
     1     1     0
     1     1     1
```

2. fullfact 函数

功能：全因子实验设计。其调用格式如下：

$dFF = \text{fullfact}(\text{levels})$

说明：给出全因子的一个参数设计。

【例 5-31】全因子实验设计示例。

```
>> dFF = fullfact([2 4])
dFF =
     1     1
     2     1
     1     2
     2     2
     1     3
     2     3
     1     4
     2     4
```

5.7 文件输入/输出

统计工具箱提供了几个必要的函数，以数据文件的形式进行数据的输入/输出。通过 Windows 标准“打开文件”和“保存文件”对话框，可以方便地操作数据文件。

5.7.1 文件输入

1. caseread 函数

功能：从文件中读示例名。其调用格式如下：

```
names = caseread(filename)
```

```
names = caseread
```

说明：读取 filename 文件的内容并返回示例名称字符串矩阵。其中，filename 是当前目录下一个文件的名称，或者是其他目录下任一文件的完整路径名。caseread 将 filename 文件中每一行看作一个单独的示例名称。

names = caseread 则显示打开文件对话框，可交互选择需要输入的文件。

【例 5-32】使用 caseread 函数参考中创建文件 Month.dat。

```
>> type months.dat
January
February
March
April
May
names = caseread('months.dat')
names =
January
February
March
April
May
```

2. casewrite 函数

功能：将串矩阵中的示例名称写入文件。其调用格式如下：

```
casewrite(strmat,filename)
```


casewrite(strmat)

说明：将 strmat 中内容写入文件。其中，filename 是当前目录下文件的名称，或者是其他目录文件的完整路径名。strmat 的每一行代表一个示例名称。casewrite 将每一示例名称写入 filename 文件的每一单独的行。

casewrite(strmat)则显示打开文件对话框，可交互选择需要输出的文件。

【例 5-33】将串矩阵中的示例名称写入文件示例。

```
>> strmat = char('January','February',...
                'March','April','May')
```

```
strmat =
```

```
January
```

```
February
```

```
March
```

```
April
```

```
May
```

```
>> casewrite(strmat,'months.dat')
```

```
type months.dat
```

```
January
```

```
February
```

```
March
```

```
April
```

```
May
```

5.7.2 文件输出

1. tblread 函数

功能：在文件系统中查询表格数据。其调用格式如下：

[data,varnames,casenames] = tblread: 显示打开文件窗口，可交互式选择需要打开的数据表格文件。文件格式为：第一行是变量名，第一列为示例名，数据则始于元素 (2, 2)。

[data,varnames,casenames] = tblread(filename): 规定了需要打开的文件 filename。filename 或是当前目录下的文件名，或是其他目录下文件的完整路径。

[data,varnames,casenames] = tblread(filename,delimiter): 还可规定表格文件数据域分隔符 (delimiter) 的格式。delimiter 可接受的值为 'tab'、'space' 和 'comma'。

tblread 返回值主要为三种类型：

(1) varnames 是包含表格文件第一行（即变量名）的字符串矩阵。

(2) casenames 是包含表格文件第一列（即示例名）的字符串矩阵。

(3) data 为对应每一变量——示例对值的数值矩阵。

【例 5-34】在文件系统中查询表格数据示例。

```
>> [data,varnames,casenames] = tblread('sat.dat')
```

```
data =
```

```
470    530
```

```
520    480
```

```
varnames =
```

```
Male
```

```
Female
```

```
casenames =  
Verbal  
Quantitative
```

2. tblwrite 函数

功能：将表格数据写入文件系统。其调用格式如下：

```
tblwrite(data,varnames,casenames)
```

```
tblwrite(data,varnames,casenames,filename)
```

```
tblwrite(data,varnames,casenames,filename,delimiter)
```

说明：上述函数语句中的参数的名称含义与 `tblread` 函数的规定相同。

【例 5-35】将表格数据写入文件系统示例。

```
>> tblwrite(data,varnames,casenames,'sattest.dat')  
>> type sattest.dat
```

	Male	Female
Verbal	470	530
Quantitative	520	480

第6章 方差分析

方差分析是实验研究中分析实验数据的重要方法，应用十分广泛。

在实际中常常要通过实验来了解各种因素对产品的性能、产量等的影响，这些性能、产量指标等统称为实验指标，而称影响实验指标的条件、原因等为因素或因子，称因素所处的不同状态为水平。各因素对实验指标的影响一般是不同的，就是一个因素的不同水平对实验指标的影响往往也是不同的。方差分析就是通过对实验数据进行分析，检验方差相同的各正态总体的均值是否相等，以判断各因素对实验指标的影响是否显著。方差分析按影响实验指标的因素的个数分为单因素方差分析、双因素方差分析和多因素方差分析，下面将对它们展开介绍。

在实验研究中，所获得的实验结果（数据）总是有差异的，即使在同一条件下重复进行实验，所得实验数据也不完全一样，引起实验数据产生差异的因素很多，这些因素对实验数据的影响程度也是不同的，有主有次，有大有小。通常由于因素变化所引起的数据差异称为条件误差，它决定了实验结果的准确度。称由于在实验过程中一系列有关因素的细小随机（偶然）的波动而形成的具有相互抵消性的误差为随机误差，它决定了实验结果的精密度。

6.1 单因素方差分析

单因素实验的方差分析有两种情况，一种是水平数相等，一种是水平数不等，这里仅讨论前者。

6.1.1 单因素方差分析问题

为了讨论方便，先给出单因素方差分析问题的一般提法。

设实验所考察的因素有 m 个水平，即 $A_1, A_2, \dots, A_i, \dots, A_m$ ，在每个水平上重复进行 r 次实验，水平 A_i 的第 j 次实验数据为 $x_{ij} (i=1, 2, \dots, m; j=1, 2, \dots, r)$ ，可得实验数据及计算表的模式见表 6-1。根据实验数据判断因素对实验是否有显著影响。

表 6-1 单因素等重复实验数据表

水平	重复实验序号						$x_{i\cdot}$	$\bar{x}_{i\cdot}$
	1	2	...	j	...	r		
A_1	x_{11}	x_{12}	...	x_{1j}	...	x_{1r}		
A_2	x_{21}	x_{22}	...	x_{2j}	...	x_{2r}		
\vdots	\vdots	\vdots		\vdots		\vdots		
A_i	x_{i1}	x_{i2}	...	x_{ij}	...	x_{ir}		
\vdots	\vdots	\vdots		\vdots		\vdots		
A_m	x_{m1}	x_{m2}	...	x_{mj}	...	x_{mr}		
总和							$x_{\cdot\cdot}$	$\bar{x}_{\cdot\cdot}$

表中:

$$x_{i.} = \sum_{j=1}^r x_{ij}, \quad \bar{x}_{i.} = \frac{1}{r} \sum_{j=1}^r x_{ij} = \frac{1}{r} x_{i.}$$

$$x_{..} = \sum_{i=1}^m \sum_{j=1}^r x_{ij}, \quad \bar{x}_{..} = \frac{1}{m} \sum_{i=1}^m \bar{x}_{i.} = \frac{1}{mr} \sum_{i=1}^m \sum_{j=1}^r x_{ij} = \frac{1}{n} x_{..}$$

$$n = mr$$

6.1.2 单因素方差分析前提条件

单因素方差分析是建立在下述假设基础上的:

- (1) 在每一个水平上实验的结果是一个随机变量 x_{ij} (i 为第 i 个水平, j 为第 j 次实验), 且服从正态分布。 $x_{i1}, x_{i2}, \dots, x_{ir}$ 是第 i 个水平的正态总体中抽出的一个简单随机样本, 样本容量为 r 。
- (2) 所有 m 个不同水平对应的 m 个正态总体的方差是相等的, 具有方差齐性, 即

$$x_{ij} \sim N(\mu_i, \sigma^2)$$

- (3) m 个总体是相互独立的, 样本与样本之间也是相互独立的。要检验的假设是

$$H_0: \mu_1 = \mu_2 = \dots = \mu_m$$

若拒绝 H_0 , 则认为至少有两个水平之间的差异是显著的, 因素 A 对实验结果有显著影响; 反之, 若接受 H_0 , 则认为因素 A 对实验结果无显著影响, 实验结果在各水平之间的不同仅仅是由于随机因素引起的。

6.1.3 单因素方差分析的步骤

1. 偏差平方和的分解

把整个实验所得的每一个实验值 x_{ij} 对其总平均 $\bar{x}_{..}$ 的偏差进行平方并求总和, 就是总的偏差平方和, 用 S_T 表示, 它反映了全面实验值之间总的波动情况, 即

$$S_T = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{..})^2 \quad (6-1)$$

将式 (6-1) 进行分解, 得

$$S_T = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{..})^2 = \sum_{i=1}^m \sum_{j=1}^r [(x_{ij} - \bar{x}_{i.}) + (\bar{x}_{i.} - \bar{x}_{..})]^2 \quad (6-2)$$

式 (6-2) 化简得

$$S_T = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{i.})^2 + \sum_{i=1}^m \sum_{j=1}^r (\bar{x}_{i.} - \bar{x}_{..})^2 = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{i.})^2 + r \sum_{i=1}^m (\bar{x}_{i.} - \bar{x}_{..})^2 \quad (6-3)$$

令

$$S_A = r \sum_{i=1}^m (\bar{x}_{i.} - \bar{x}_{..})^2 \quad (6-4)$$

它是各条件 (水平) 下的平均数与总平均数的偏差平方和, 反映了因素 A 的水平变化引起的波动, 称为组间偏差平方和或因因素平方和。

令

$$S_e = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{i.})^2 \quad (6-5)$$

它是各条件（水平）下的实验值与该条件下平均值偏差的平方和，反映了随机误差引起的波动，称为组内偏差平方和或误差平方和。

于是有

$$S_T = S_A + S_e \quad (6-6)$$

由图 6-1 (a) 也可直观地看出相应的结论。由于 $x_{ij} \sim N(\mu_i, \sigma^2)$ ，由图 6-1 (a) 知

$$x_{ij} = \mu_i + \varepsilon_{ij} \quad (6-7)$$

其中， ε_{ij} 为随机误差（组内误差）。

由图 6-1 (b) 知

$$\mu_i = \mu + \alpha_i \quad (6-8)$$

其中， α_i 为条件误差（组间误差）。

由式 (6-7) 及式 (6-8)，有

$$x_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

即

$$x_{ij} - \mu = \alpha_i + \varepsilon_{ij}$$

这说明任意观测数据与总平均值的误差，都可以分解成条件误差与随机误差的和。

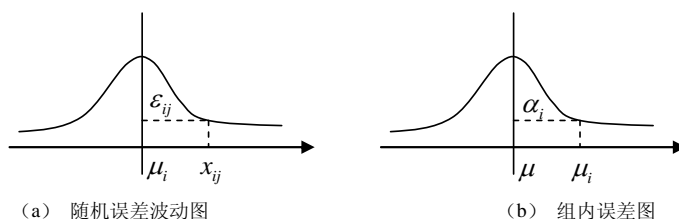


图 6-1 误差图

2. 偏差平方和的简化计算

在实际运用中，为计算简便，常用下列简便算法求 S_T 、 S_A 和 S_e ：

$$S_T = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{..})^2 = \sum_{i=1}^m \sum_{j=1}^r x_{ij}^2 - \frac{1}{n} x_{..}^2$$

令

$$Q_T = \sum_{i=1}^m \sum_{j=1}^r x_{ij}^2, \quad C_T = \frac{1}{n} x_{..}^2 \quad (C_T \text{ 为修正项})$$

则

$$S_T = Q_T - C_T$$

同样，有

$$S_A = r \sum_{i=1}^m (\bar{x}_{i.} - \bar{x}_{..})^2 = \frac{1}{r} \sum_{i=1}^m x_{i.}^2 - \frac{1}{n} x_{..}^2$$

令

$$Q_A = \frac{1}{r} \sum_{i=1}^m x_i^2$$

则

$$S_A = Q_A - C_T$$

一般是先求出 S_T 和 S_A ，再利用

$$S_e = S_T - S_A$$

求出 S_e 。计算可在数据计算表（表 6-1）上进行。

3. 平均偏差平方和

偏差平方和的大小与参与求和的项数有关，为了比较 S_A 与 S_e 的大小，应消除求和项数的影响，比较它们的平均值，从数学上的理论推导知道， S_A 与 S_e 的平均值，不是 S_A 与 S_e 分别除以相应的参与求和的项数，而应除以它们的自由度，下面分别讨论 S_T 、 S_A 和 S_e 的自由度 f_T 、 f_A 和 f_e 。

式 $S_T = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{..})^2$ 中有 $mr = n$ 个数据 x_{ij} ，存在一个线性约束 $\sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{..}) = 0$ ，故 S_T 的自由度为 $f_T = n - 1$ 。

式 $S_A = r \sum_{i=1}^m (\bar{x}_{i.} - \bar{x}_{..})^2$ 中，有 m 个数据 $\bar{x}_{i.}$ ，存在一个线性约束 $\sum_{i=1}^m (\bar{x}_{i.} - \bar{x}_{..}) = 0$ ，故 S_A 的自由度为 $f_A = m - 1$ 。

式 $S_e = \sum_{i=1}^m \sum_{j=1}^r (x_{ij} - \bar{x}_{i.})^2$ 中，有 $mr = n$ 个数据 x_{ij} ，存在 m 个线性约束 $\sum_{j=1}^r (x_{ij} - \bar{x}_{i.}) = 0 (i = 1, 2, \dots, m)$ ，故 S_e 的自由度为 $f_e = mr - m = n - m$ 。

显然有

$$f_T = f_A + f_e \quad (6-9)$$

式 (6-9) 称为偏差平方和自由度分解公式。因为总自由度 $f_T = n - 1$ 是总的的数据个数减 1，而组间自由度 $f_A = m - 1$ 是因素的水平数减 1，都很好计算，所以一般先求出 f_T 和 f_A ，再利用

$$f_e = f_T - f_A$$

求出组内自由度 f_e ，于是可求出平均值

$$V_A = \frac{S_A}{f_e} \quad (6-10)$$

$$V_e = \frac{S_e}{f_e} \quad (6-11)$$

4. 显著性检验

若 H_0 为真，即 $\mu_1 = \mu_2 = \dots = \mu_m$ ，则全体样本可看作是来自同一正态总体 $N(\mu_i, \sigma^2)$ 。因为，

$\frac{S_T}{n-1}$ 、 $\frac{S_A}{m-1}$ 、 $\frac{S_e}{n-m}$ 都是总体方差 σ^2 公式的无偏估计值，且当原假设 H_0 成立时， S_A 和 S_e 分别是自由度为 $m-1$ 、 $n-m$ 的 χ^2 变量，所以统计量为

$$F = \frac{S_A/(m-1)}{S_e/(n-m)} = \frac{V_A}{V_e} \sim F(m-1, n-m) \quad (6-12)$$

显然, F 应接近于 1。如果 F 值比 1 大得多, 即 V_A 明显地大于 V_e , 就有理由认为原假设公式不成立。表明 S_A 中不仅包括随机误差, 而且包含因素 A 水平变动引起的数据波动 (称为因素误差), 即因素 A 对实验结果影响显著。这种比较方差大小来判断原假设 H_0 是否成立的方法, 就是方差分析名称的由来。

现在问题是, F 值大到多大, 可以认为实验结果的差异主要由因素水平的改变引起的; 小到多小, 可以认为实验结果的差异主要是由实验误差引起的, 这就需要一个比较的标准。对于给定的信度 α , 可查表得出临界值 F_α , 将由样本值算得的 F 的值 F_0 与 F_α 比较, 若 $F_0 < F_\alpha$, 则接受原假设即认为因素 A 对实验结果无显著影响。

对于检验水平 α 的选取, 视具体情况而定, 通常取 $\alpha=0.01$ 和 $\alpha=0.05$, 从 F 分布表上查出 $F_{0.01}$ 和 $F_{0.05}$ 。若 $F_0 \geq F_\alpha$, 判定因素 A 为高度显著, 记为 “**”; 若 $F_{0.05} \leq F_0 \leq F_{0.01}$, 判定因素 A 为显著, 记为 “*”; 若 $F_0 < F_{0.05}$, 则判定因素 A 为不显著, 不作标记。

6.1.4 单因素方差分析的 MATLAB 实现

在 MATLAB 统计工具箱中提供 anova1 函数实现单因素方差。其调用格式如下:

```
p=anova1(X)
p=anova1(X,group)
p=anova1(X,group,displayopt)
[p,table]=anova1(...)
[p,table,stats]=anova1(...)
```

说明: 方差分析法, 是基于统计数据的总变动中因素引起的变动成分与随机误差成分的比较; 是基于不多的统计数据, 定量地分析一个或多个因素对某个应变量的影响和作用的显著性。其前提条件是在各因素作用下应变变量分布的正态性和等方差性。

$p=anova1(X)$ 对样本 X 中的两列或多列数据进行均衡的单因素方差分析, 以比较各列的均值。函数返回 “零假设” (X 中各列的均值相同) 成立的概率值。如果概率值接近于零, 则零假设值得怀疑, 表明各列的均值事实上是不同的。 $p=anova1(X,group)$ 对样本 X 中由矢量 $group$ 索引的两组或多组数据进行单因素方差分析以比较各列的均值。输入参数 $group$ 标明矢量 X 中相应元素的组别。 $group$ 中的值为整数, 最大值为需要比较的不同组的数量, 最小值为 1。每组中至少应有一个元素, 但并不要求每组中元素个数相同, 因此适合于数据不均衡的情况。用以决定结果是否具有统计上的显著性, 并且概率值的大小由用户选择。 $anova1$ 同时还显示一张表和一幅图。表为标准的 ANOVA 表, 表中将 X 中数据的变化分成两部分:

- (1) 由各列均值差异而产生的变化。
- (2) 由各列的数据与其均值间的差异而导致的变化。

ANOVA 表具有 5 列数据:

- (1) 第一列标明数据源。
- (2) 第二列给出相应数据源的均方和 (SS)。
- (3) 第三列给出相应数据源的自由度 df。
- (4) 第四列给出均方值 p , 即比率 SS/df 。
- (5) 第五列给出 F 统计量。

p 值是 F 的函数 ($fcdF$)。随着 F 的增加, p 值减少。在 box 图中, 各列数据图的中心线若出现较大差异, 则对应的 F 值较大、 p 值较小。

【例 6-1】X 中的五列数据分别为 1~5 的常数与均值为 0、标准差为 1 的正态随机干扰量之和。

```
>> X = meshgrid(1:5)
X =
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
>> X = X + normrnd(0,1,5,5)
X =
    -0.5062    2.4434    2.4922    5.2424    5.1825
     0.5554    2.3919    2.6794    2.9333    3.4349
     0.8441    0.7493    3.0125    4.9337    4.9155
     1.2761    1.0520    -0.0292    4.3503    6.6039
     0.7388    1.2589    2.5430    3.9710    5.0983
>> p = anova1(X)
```

运行程序，输出如下，效果如图 6-2 及图 6-3 所示。

```
p =
    1.4223e-006
```

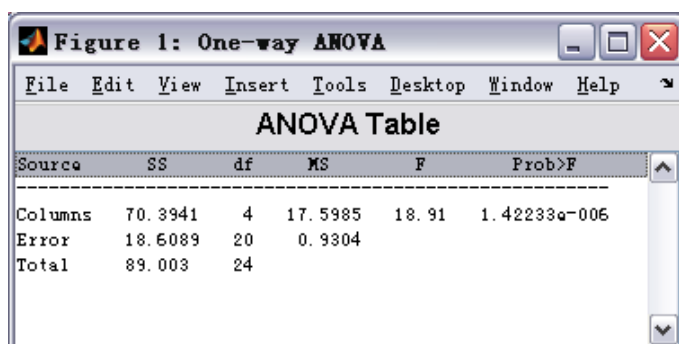


图 6-2 ANOVA 表

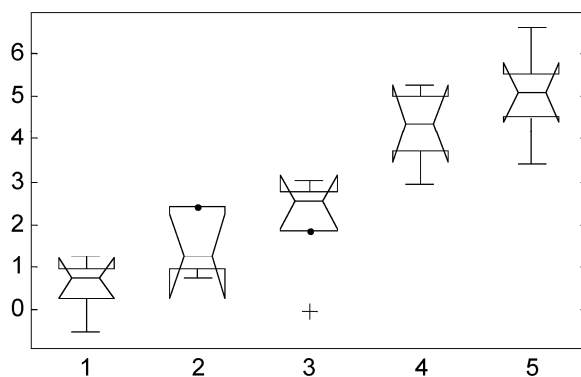


图 6-3 box 效果图

由计算结果可知，观察所提供的 X 的随机数据样本，给出其各列均值相同结论的概率小于 10^{-6} 。

【例 6-2】下面的例子源自结构横梁材料的应力研究。在每 1% 英尺^①材料上施以 3000N 的力，观察横梁的挠曲，观察结果保存在矢量 `strength` 中。强韧的横梁挠曲较少。建筑工程师进行此项研究的目的在于，确定钢质横梁的应力强度是否等同于其他两种更昂贵的合金的应力强度。在矢量 `alloy` 中，钢材编码为 1，其他两种合金材料的编码分别为 2 和 3。

```
>> strength = [82 86 79 83 84 85 86 87 74 82 ...
               78 75 76 77 79 79 77 78 82 79];
alloy = {'st','st','st','st','st','st','st','st',...
         'al1','al1','al1','al1','al1','al1',...
         'al2','al2','al2','al2','al2','al2'};
```

结果表明，三种材料的应力强度是明显不同的。box 图显示出钢材的挠曲比其他两种合金的挠曲程度更大。

```
p = anova1(strength,alloy)
p =
    1.5264e-004
```

运行程序，效果如图 6-4 及图 6-5 所示。

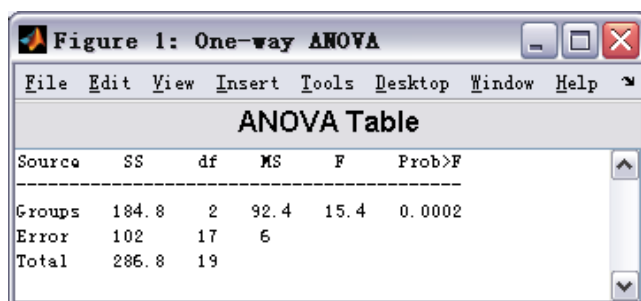


图 6-4 ANOVA 表

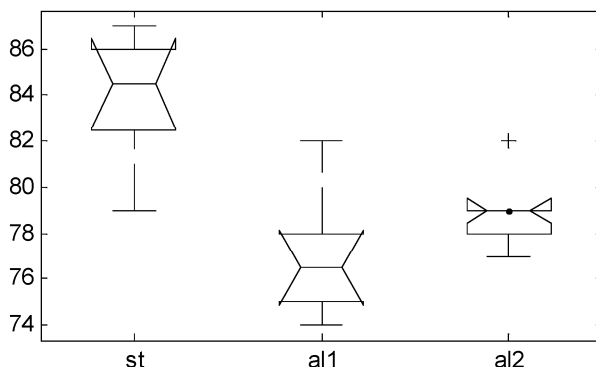


图 6-5 box 效果图

【例 6-3】一位英语教师想检查 3 种不同的教学方法的效果，为此随机选取 24 名学生并把他们分成 3 组，相应地用 3 种方法教学。一段时间后，这位教师对这 24 名学生进行统考，统考成绩见表 6-2。试问在 0.05 显著性水平下，这 3 种教学方法有无显著性差异？作方差齐性检验。

① 1 英尺=0.3048m。

表 6-2 英语成绩表

方法	学习成绩								
A ₁	73	66	89	82	43	80	63		
A ₂	88	78	91	76	85	84	80	96	
A ₃	68	79	71	71	87	68	59	76	80

假设 $H_0: \sigma_1^2 = \sigma_2^2 = \sigma_3^2$ ，即 3 个变量的方差相等。按照上述结论，分别求得例 6-2 中检验统计量 B 的值和本题的拒绝域，经过比较得出结论。

```
>> %MATLAB 数据处理(4)
clear all;
y=[73 66 89 82 43 80 63 88 78 91 76 85 94 80 96 68 79 71 71 87 68 59 76 80];
alpha=0.05;
m1=7;m2=8;m3=9; %各总体的样本容量
n=m1+m2+m3;
r=3;
SSE=2.3404e+003; %引用 MATLAB 数据处理(1)中结果
n=m1+m2+m3;
fE=n-r;
c=(1/(m1-1)+1/(m2-1)+1/(m3-1)-1/fE)/(3*(r-1))+1;
s1=var(y(1:m1));
s2=var(y((m1+1):(m1+m2)));
s3=var(y((n-m3+1):n));
chi2EST=(fE*log(SSE/fE)-(m1-1)*log(s1)-(m2-1)*log(s2)-(m3-1)*log(s3))/c;
LJZ=chi2cdf(1-alpha,r-1);
p=1-chi2cdf(chi2EST,r-1);
if chi2EST>LJZ
    h=1;
else
    h=0;
end
alpha,h,p,chi2EST,LJZ
```

运行程序输出如下：

```
alpha =    0.0500
h =      1
p =    0.1330
chi2EST =    4.0348
LJZ =    0.3781
```

计算结果表明，在 0.05 显著性水平下， $h=1$ 、 $p>\alpha$ 不能拒绝原假设，即认为 3 种教学方法下学生的英语成绩这 3 个变量方差相等。

6.2 双因素方差分析

双因素试验方差分析的基本思想和方法与单因素试验方差分析相似，前提条件仍然是独立、方差具有齐性、正态。所不同的是在双因素试验中，有可能出现交互作用。按照是否进行

重复试验, 双因素方差分析又分为两种: 双因素等重复试验和双因素无重复试验。

6.2.1 双因素等重复试验的方差分析

1. 双因素等重复试验分析

设有两个因素 A 、 B 作用于试验的指标。因素 A 有 r 个水平, 即 A_1, A_2, \dots, A_r ; 因素 B 有 s 个水平, 即 B_1, B_2, \dots, B_s 。现对因素 A 、 B 的每对水平组合 (A_i, B_j) , $(i=1, 2, \dots, r; j=1, 2, \dots, s)$ 都做 $t(t \geq 2)$ 次试验 (等重复试验), 得到表 6-3 所列的结果。

表 6-3 双因素等重复试验

因素 B 因素 A	B_1	B_2	...	B_s
A_1	X_{111}	X_{121}	...	X_{1s1}
	X_{112}	X_{122}	...	X_{1s2}
	\vdots	\vdots	\vdots	\vdots
	X_{11r}	X_{12r}	...	X_{1sr}
A_2	X_{211}	X_{221}	...	X_{2s1}
	X_{212}	X_{222}	...	X_{2s2}
	\vdots	\vdots	\vdots	\vdots
	X_{21r}	X_{22r}	...	X_{2sr}
\vdots	\vdots	\vdots	\vdots	\vdots
A_r	X_{r11}	X_{r21}	...	X_{rs1}
	X_{r12}	X_{r22}	...	X_{rs2}
	\vdots	\vdots	\vdots	\vdots
	X_{r1r}	X_{r2r}	...	X_{rst}

设 $X_{ijk} \sim N(\mu_{ij}, \sigma^2)$ ($i=1, 2, \dots, r; j=1, 2, \dots, s; k=1, 2, \dots, t$), 且各 X_{ijk} 独立, $X_{ijk} - \mu_{ij} = \varepsilon_{ijk} \sim N(0, \sigma^2)$, 再引入记号。

$$\mu = \frac{1}{rs} \sum_{i=1}^r \sum_{j=1}^s \mu_{ij}$$

$$\mu_{.i} = \frac{1}{s} \sum_{j=1}^s \mu_{ij} \quad (i=1, 2, \dots, r)$$

$$\mu_{.j} = \frac{1}{r} \sum_{i=1}^r \mu_{ij} \quad (j=1, 2, \dots, s)$$

$$\alpha_i = \mu_{.i} - \mu \quad (i=1, 2, \dots, r)$$

$$\beta_j = \mu_{.j} - \mu \quad (j=1, 2, \dots, s)$$

$$\gamma_{ij} = \mu_{ij} - \mu_{.i} - \mu_{.j} + \mu \quad (i=1, 2, \dots, r; j=1, 2, \dots, s)$$

式中: μ 为总平均; α_i 为水平 A_i 的效应; β_j 为水平 B_j 的效应; γ_{ij} 为水平 A_i 和水平 B_j 的交互效应, 这是由 A_i 、 B_j 搭配起来联合起作用而引起的。于是有

$$X_{ijk} = \mu_{ij} + \varepsilon_{ijk} = \mu + (\mu_i - \mu) + (\mu_j - \mu) + (\mu_{ij} - \mu_i - \mu_j + \mu) + \varepsilon_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk}$$

则

$$\begin{cases} X_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk} \\ \varepsilon_{ijk} \sim N(0, \sigma^2), \text{各 } \varepsilon_{ijk} \text{ 独立} \\ i = 1, 2, \dots, r; j = 1, 2, \dots, s; k = 1, 2, \dots, t \\ \sum_{i=1}^r \alpha_i = 0, \sum_{j=1}^s \beta_j = 0, \sum_{i=1}^r \gamma_{ij} = 0, \sum_{j=1}^s \gamma_{ij} = 0 \end{cases}$$

这就是双因素试验方差分析的数学模型。这一模型要检验以下三个假设：

$$\begin{cases} H_{01} : \alpha_1 = \alpha_2 = \dots = \alpha_r = 0 \\ H_{11} : \alpha_1, \alpha_2, \dots, \alpha_r \text{ 不全为零} \\ H_{02} : \beta_1 = \beta_2 = \dots = \beta_s = 0 \\ H_{12} : \beta_1, \beta_2, \dots, \beta_s \text{ 不全为零} \\ H_{03} : \gamma_{11} = \gamma_{12} = \dots = \gamma_{rs} = 0 \\ H_{13} : \gamma_{11}, \gamma_{12}, \dots, \gamma_{rs} \text{ 不全为零} \end{cases}$$

记

$$\begin{aligned} \bar{X} &= \frac{1}{rst} \sum_{i=1}^r \sum_{j=1}^s \sum_{k=1}^t X_{ijk} \\ \bar{X}_{ij.} &= \frac{1}{t} \sum_{k=1}^t X_{ijk} \quad (i = 1, 2, \dots, r; j = 1, 2, \dots, s) \\ \bar{X}_{i..} &= \frac{1}{st} \sum_{j=1}^s \sum_{k=1}^t X_{ijk} \quad (i = 1, 2, \dots, r) \\ \bar{\mu}_j &= \frac{1}{st} \sum_{i=1}^r \sum_{k=1}^t X_{ijk} \quad (j = 1, 2, \dots, s) \end{aligned}$$

记

$$\begin{aligned} SST &= \sum_{i=1}^r \sum_{j=1}^s \sum_{k=1}^t (X_{ijk} - \bar{X})^2 \\ SSE &= \sum_{i=1}^r \sum_{j=1}^s \sum_{k=1}^t (X_{ijk} - \bar{X}_{ij.})^2 \\ SSA &= st \sum_{i=1}^r (\bar{X}_{i..} - \bar{X})^2 \\ SSB &= rt \sum_{j=1}^s (\bar{X}_{.j.} - \bar{X})^2 \\ SSA \times B &= t \sum_{i=1}^r \sum_{j=1}^s (\bar{X}_{ij.} - \bar{X}_{i..} - \bar{X}_{.j.} + \bar{X})^2 \end{aligned}$$

式中：SSE 为误差平方和；SSA 为因素 A 的效应平方和；SSB 为因素 B 的效应平方和；SSA × B 为因素 A、B 的交互效应平方和。则有

$$SST = SSA + SSB + SSA \times B + SSE$$

可以证明, 当 $H_{01}: \alpha_1 = \alpha_2 = \cdots = \alpha_r = 0$ 为真时, 有

$$F_A = \frac{SSA/(r-1)}{SSE/rs(t-1)} \sim F(r-1, rs(t-1))$$

在显著水平为 α 时, H_{01} 的拒绝域为

$$F_B = \frac{SSA/(r-1)}{SSE/rs(t-1)} \geq F(r-1, rs(t-1))$$

类似可得, 在显著水平为 α 时, H_{02} 的拒绝域为

$$F_B = \frac{SSA/(s-1)}{SSE/rs(t-1)} \geq F_\alpha(s-1, rs(t-1))$$

在显著水平为 α 时, H_{03} 的拒绝域为

$$F_{A \times B} = \frac{SSA \times B / ((r-1)(s-1))}{SSE/rs(t-1)} \geq F_\alpha((r-1), rs(t-1))$$

对于检验水平 α 的选取, 视具体情况而定。通常取 $\alpha = 0.01$ 和 $\alpha = 0.05$, 从 F 分布表上查出 $F_{0.01}$ 和 $F_{0.05}$ 。若 $F \geq F_{0.01}$, 判定因素为高度显著; 若 $F_{0.05} \leq F < F_{0.01}$, 判定因素为显著; 若 $F < F_{0.05}$, 则判定因素为不显著。

得到的双因素等重复试验的方差分析表见表 6-4。

表 6-4 双因素等重复试验方差分析表

方差来源	平方和	自由度	均方差	F 值
因素 A	SSA	$f_A = r - 1$	$\frac{SSA}{f_A}$	$F_A = \frac{SSA/(r-1)}{SSE/rs(t-1)}$
因素 B	SSB	$f_B = s - 1$	$\frac{SSB}{f_B}$	$F_B = \frac{SSA/(r-1)}{SSE/rs(t-1)}$
交互效应 A×B	SSAB	$f_{AB} = (r-1)(s-1)$	$\frac{SSAB}{f_{AB}}$	$F_{A \times B} = \frac{SSAB}{f_{AB}} / \frac{SSE}{f_E}$
误差 E	SSE	$f_E = rs(n-1)$	$\frac{SSE}{f_E}$	
总和 T	SST	$f_T = rsn - 1$		

2. MATLAB 中双因素等重复试验的方差分析实现

在 MATLAB 统计工具箱中提供了 anova2 函数进行双因素等重复试验的方差分析。其调用格式如下:

```
p = anova2(X, reps)
p = anova2(X, reps, displayopt)
[p, table] = anova2(...)
[p, table, stats] = anova2(...)
```

说明：双因素方差分析是一种两因素、多水平析因试验数据的统计分析方法，目的在于确认来自不同组的数据是否具有相同的均值。

假设一个汽车制造公司有两个厂，都分别制造 3 种汽车。考虑汽车的燃气里程（每升汽油所跑的里程数）随汽车种类不同和工厂不同而变化的情况。由于工厂制造方法的差异，使燃气里程有总体的差别；而由于设计规定的差异，不同种类汽车的燃气里程也可能不同。另外，制造方法和设计规定二者也可能存在综合效应，而影响汽车的燃气里程。因此，除非对工厂和汽车种类相结合进行观察，否则不可能观测到交互作用。

双因素方差分析是处理这种问题的典型方法。首先建立问题的数学模型如下：

$$y_{ijk} = \mu + \alpha_j + \beta_i + \gamma_{ij} + \varepsilon_{ijk}$$

式中： y_{ijk} 为观测值矩阵； μ 为样本总均值（常数均值）； α_j 为列元素为组均值的矩阵（各行 α 总和为 0）； β_i 为行元素为组均值的矩阵（各列 β 总和为 0）； γ_{ij} 为交互作用项（矩阵）（各行 γ 的总和为 0）； ε_{ijk} 为随机干扰矩阵。

返回“零假设”（列数据的均值与数据的均值相同）成立的概率值 p 。如果概率值接近于零，则零假设值得怀疑。用于决定结果是否具有统计上的显著性，并且概率值的大小由用户选择。通常认为，如果 p 值小于 0.05 或 0.1，则结果较显著。同时也显示一个标准方差分析表（ANOVA 表），其中，按照 `reps` 参数值将 X 中数据的变化情况分成 3 部分或 4 部分：

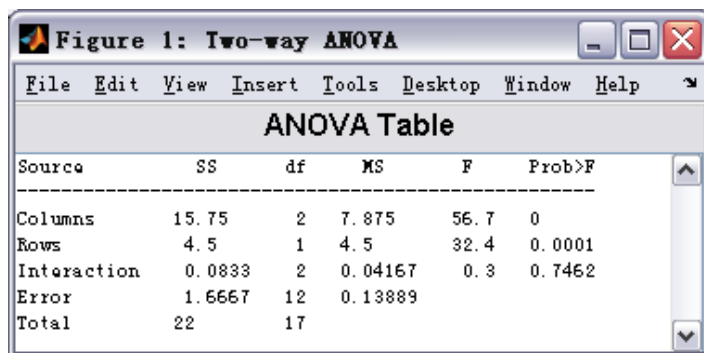
- (1) 由各列均值差异而产生的变化。
- (2) 由各行均值差异而产生的变化。
- (3) 由列和行因素的交互作用而导致的变化（如果 `reps` 值大于其默认值 1）。
- (4) 其他因素。

p 值是 F 的函数（`fcdf`），随着 F 值的增加 p 值减少。

【例 6-4】双因素方差分析。

```
>> load popcorn
popcorn
popcorn =
    5.5000    4.5000    3.5000
    5.5000    4.5000    4.0000
    6.0000    4.0000    3.0000
    6.5000    5.0000    4.0000
    7.0000    5.5000    5.0000
    7.0000    5.0000    4.5000
>> p = anova2(popcorn,3)
p =
    0.0000    0.0001    0.7462
```

运行程序，输出的 ANOVA 表如图 6-6 所示。



Source	SS	df	MS	F	Prob>F
Columns	15.75	2	7.875	56.7	0
Rows	4.5	1	4.5	32.4	0.0001
Interaction	0.0833	2	0.04167	0.3	0.7462
Error	1.6667	12	0.13889		
Total	22	17			

图 6-6 ANOVA 表

由 MATLAB 生成的方差分析表与所列的方差分析表的行列因素在方差分析表中的位置不一样，可以通过编程改变熟悉的形式。

其源程序代码如下：

```
function table=anova2c(a, reps)
if nargin<2
    error('请输入单元的行数');
end
[m,n]=size(a);
if mod(m,reps)
    error('矩阵 a 的行数必须是 reps 的倍数');
end
alpha=[0.05,0.01];
format short g
[p,b]=anova2(a,reps,'off');    %调用方差分析表函数
table=b;
table(2,:)=b(3,:);
table(3,:)=b(2,:);
table(1,1:7)={'方差来源','偏差平方和','自由度','方差','F 值','Fa','显著性'};
table(2,6,1)={'因素 a(行间)','因素 b(列间)','交互作用','误差','总和'};
f1=finv(1-alpha,table{2,3},table{5,3});
f2=finv(1-alpha,table{3,3},table{5,3});
f3=finv(1-alpha,table{4,3},table{5,3});
fa=table{2,5};
fb=table{3,5};
fab=table{4,5};
table{2,6}=[num2str(f1(1)),',',num2str(f1(2))];
table{3,6}=[num2str(f2(1)),',',num2str(f2(2))];
table{4,6}=[num2str(f3(1)),',',num2str(f3(2))];
if fa>=f1(2)
    table{2,7}='高度显著';
elseif (fa>=f1(1))&(fa<f1(2))
    table{2,7}='显著';
else
    table{2,7}='不显著';
end
```

```

if fb>=f2(2)
    table{3,7}='高度显著';
elseif (fb>=f2(1))&(fb<f2(2))
    table{3,7}='显著';
else
    table{3,7}='不显著';
end
if fab>=f3(2)
    table{4,7}='高度显著';
elseif (fab>=f3(1))&(fab<f3(2))
    table{4,7}='显著';
else
    table{4,7}='不显著';
end

```

这一程序的输出参数是一个单元数组，它的数据可以生成双因素等重复试验的方差分析表。

【例 6-5】在某橡胶配方中，考虑 3 种不同的促进剂，4 种不同成分的氧化锌，同样的配方重复一次，测得 300% 的定伸强力见表 6-5。试问氧化锌、促进剂以及它们的交互作用对定伸强力有无显著影响？

表 6-5 不同的氧化锌、促进剂下定伸强力

因素 A \ 因素 B				
	B_1	B_2	B_3	B_4
A_1	31	37	35	38
	34	36	37	39
A_2	33	34	38	42
	35	35	34	43
A_3	33	39	33	38
	34	36	40	37

其实现的 MATLAB 程序代码如下：

```

>> clear all;
a=[31 37 35 38;34 36 37 39;...
   33 34 38 42;35 35 34 43;...
   33 39 33 38;34 36 40 37];
table=anova2c(a,2)

```

运行程序，输出结果如下：

```

table =
Columns 1 through 5
'方差来源'    '偏差平方和'    '自由度'    '方差'    'F 值'
'因素 a(行间)' [ 3.0833] [ 2] [1.5417] [0.38144]
'因素 b(列间)' [ 114.46] [ 3] [38.153] [ 9.4399]
'交互作用' [ 36.917] [ 6] [6.1528] [ 1.5223]
'误差' [ 48.5] [ 12] [4.0417] [ ]
'总和' [ 202.96] [ 23] [ ] [ ]

```


Columns 6 through 7

'Fa'	'显著性'
'3.8853;6.9266'	'不显著'
'3.4903;5.9525'	'高度显著'
'2.9961;4.8206'	'不显著'
□	□
□	□

从运行的结果可以看出：氧化锌、促进剂对定伸强力有的影响是高度显著的；但两者的交互作用对定伸强力的影响不显著。

6.2.2 双因素无重复试验的方差分析

1. 双因素无重复试验分析

1) 偏差平方和的分解

为了构造检验用统计量，仿照单因素方差分析方法，先对偏差平方和进行分解，即

$$\begin{aligned}
 S_T &= \sum_{i=1}^a \sum_{j=1}^b (x_{ij} - \bar{x}_{..})^2 \\
 &= \sum_{i=1}^a \sum_{j=1}^b (\bar{x}_{i.} - \bar{x}_{..})^2 + \sum_{i=1}^a \sum_{j=1}^b (\bar{x}_{.j} - \bar{x}_{..})^2 + \sum_{i=1}^a \sum_{j=1}^b (x_{ij} - \bar{x}_{i.} - \bar{x}_{.j} + \bar{x}_{..})^2 \\
 &= b \sum_{i=1}^a (\bar{x}_{i.} - \bar{x}_{..})^2 + a \sum_{j=1}^b (\bar{x}_{.j} - \bar{x}_{..})^2 + \sum_{i=1}^a \sum_{j=1}^b (x_{ij} - \bar{x}_{i.} - \bar{x}_{.j} + \bar{x}_{..})^2
 \end{aligned} \quad (6-13)$$

令

$$S_A = b \sum_{i=1}^a (\bar{x}_{i.} - \bar{x}_{..})^2 \quad (6-14)$$

式中： S_A 为因素 A 各水平间，即各行间的偏差平方和，反映了因素 A 的实验结果的影响。令

$$S_B = a \sum_{j=1}^b (\bar{x}_{.j} - \bar{x}_{..})^2 \quad (6-15)$$

式中： S_B 为因素 B 各水平间，即各列间的偏差平方和，反映了因素 B 对实验结果的影响。

令

$$S_e = \sum_{i=1}^a \sum_{j=1}^b (x_{ij} - \bar{x}_{i.} - \bar{x}_{.j} + \bar{x}_{..})^2 \quad (6-16)$$

式中： S_e 为误差偏差平方和，即组内偏差平方和，反映了实验误差的大小。

于是式 (6-13) 可记为

$$S_T = S_A + S_B + S_e \quad (6-17)$$

2) 偏差平方和的简化计算

$$S_T = \sum_{i=1}^a \sum_{j=1}^b (x_{ij} - \bar{x}_{..})^2 = \sum_{i=1}^a \sum_{j=1}^b x_{ij}^2 - \frac{1}{n} x_{..}^2 = Q_T - C_T \quad (6-18)$$

$$S_A = b \sum_{i=1}^a (\bar{x}_{i.} - \bar{x}_{..})^2 = \frac{1}{b} \sum_{i=1}^a x_{i.}^2 - \frac{1}{n} x_{..}^2 = Q_A - C_T \quad (6-19)$$

$$S_B = a \sum_{j=1}^b (\bar{x}_{.j} - \bar{x}_{..})^2 = \frac{1}{a} \sum_{j=1}^b x_{.j}^2 - \frac{1}{n} x_{..}^2 = Q_B - C_T \quad (6-20)$$

$$S_e = S_T - S_A - S_B \quad (6-21)$$

3) 计算自由度和方差

S_T 的自由度为

$$f_T = ab - 1 = n - 1$$

S_A 的自由度为

$$f_A = a - 1$$

S_B 的自由度为

$$f_B = b - 1$$

S_e 的自由度为

$$f_e = f_T - f_A - f_B = (a - 1)(b - 1)$$

将各偏差平方和除以相应的自由度, 可求得各行间、各列间和误差的方差如下。

行间方差为

$$V_A = \frac{S_A}{f_A} = \frac{S_A}{a - 1} \quad (6-22)$$

列间方差为

$$V_B = \frac{S_B}{f_B} = \frac{S_B}{b - 1} \quad (6-23)$$

误差方差为

$$V_e = \frac{S_e}{f_e} = \frac{S_e}{(a - 1)(b - 1)} \quad (6-24)$$

4) 显著性检验

数学上可以证明: 假设 H_{01} 为真时, 统计量

$$F_A = \frac{V_A}{V_e} = \frac{S_A / (a - 1)}{S_e / ((a - 1)(b - 1))} \sim F[(a - 1), (a - 1)(b - 1)] \quad (6-25)$$

假设 H_{02} 为真时, 统计量

$$F_B = \frac{V_B}{V_e} = \frac{S_B / (b - 1)}{S_e / ((a - 1)(b - 1))} \sim F[(b - 1), (a - 1)(b - 1)] \quad (6-26)$$

因此, 利用 F_A 与 F_B 就可以分别对因素 A 和 B 作用的显著性进行检验。对于给定的显著性水平 α , 在相应的自由度下查出 $F_{A\alpha}$ 和 $F_{B\alpha}$, 若 $F_A \geq F_{A\alpha}$, 拒绝 H_{01} , 反之, 则接受 H_{01} ; 若 $F_B \geq F_{B\alpha}$, 则拒绝 H_{02} , 反之, 则接受 H_{02} 。

2. MATLAB 中双因素无重复试验的方差分析实现

对于双因素无重复试验的方差分析, 也可以通过修改由 MATLAB 生成的方差分析表得到。由于由 MATLAB 生成的方差分析表先列出的是列因素, 而通常所用的方差分析表中先列出的是行因素, 所以在编程时, 需要把 MATLAB 生成的方差分析表中行因素与列因素的位置交换, 以符合通常的形式。

其源程序代码如下:

```

function table=anova2s(a)
alpha=[0.05,0.01];
format short g
[p,b]=anova2(a,1,'off');
table=b;
table(2,:)=b(3,:); %交换行列的位置
table(3,:)=b(2,:);
table(1,1:7)={'方差来源','偏差平方和','自由度','方差','F 值','Fa','显著性'};
table(2:5,1)={'因素 a(行间)';'因素 b(列间)';'误差';'总和'};
f1=finv(1-alpha,table{2,3},table{4,3});
f2=finv(1-alpha,table{3,3},table{4,3});
fa=table{2,5};
fb=table{3,5};
table{2,6}=[num2str(f1(1))',';',num2str(f1(2))];
table{3,6}=[num2str(f2(1))',';',num2str(f2(2))];
if fa>=f1(2)
    table{2,7}='高度显著';
elseif (fa>f1(1))&(fa<=f1(2))
    table{2,7}='显著';
else
    table{2,7}='不显著';
end
if fb>f2(2)
    table{3,7}='高度显著';
elseif (fb>f2(1))&(fb<=f2(2))
    table{3,7}='显著';
else
    table{3,7}='不显著';
end
end

```

程序的返回参数是一个单元数组，由它可以生成方差分析表。

【例 6-6】将落叶松苗木栽在 4 块不同苗床上，每块苗床上苗木又分别使用 3 种不同的肥料以观察肥效差异，一年后于每一苗床的各施肥小区内用重复抽样方式各抽取苗木若干株，测其平均高，得到的数据见表 6-6。

表 6-6 不同肥料、不同苗床的苗高平均值

因素 B 因素 A	B_1	B_2	B_3	B_4
A_1	49	44	46	48
A_2	61	55	57	79
A_3	52	48	59	47

试问不同肥料与不同苗床对苗高生长有无显著影响？

其实现的 MATLAB 程序代码如下：

```

>> clear all;
a=[49    44    46    48;...
    61    55    57    79;...

```

```
52 48 59 47];
table=anova2s(a)
```

运行程序，输出如下：

```
table =
Columns 1 through 5
'方差来源'      '偏差平方和'      '自由度'      '方差'      'F 值'
'因素 a(行间)'  [    558.5]      [    2]      [279.25]      [ 4.9063 ]
'因素 b(列间)'  [   122.25]      [    3]      [ 40.75]      [0.71596]
'误差'          [   341.5]      [    6]      [56.917]      [    ]
'总和'          [  1022.3]      [   11]      [    ]      [    ]
Columns 6 through 7
'Fa'              '显著性'
'5.1433;10.9248'  '不显著'
'4.7571;9.7795'   '不显著'
                []      []
                []      []
```

从运行结果可以看出：不同肥料与不同苗床对苗高生长的影响都不显著。

6.3 多因素方差分析

在 MATLAB 中提供了 `anovan` 函数实现 N 因素方差分析函数。其调用格式如下：

`p = anovan(y,group)`: 比较 `y` 中对应于 N 个不同因子的观察值的均值。因子和因子水平由单元数 `group` 指定。`group` 中 N 个单元中的每一个包含一系列因子水平，确定相对于 N 个因子中某一个 X 的观察值。每个单元中的列表可以是向量、字符数组或字符串单元数组，并且必须与 X 具有相同的元素个数。`p` 为零假设的概率，当小于 0.05 或 0.01 时，一般认为可以拒绝零假设。

`p = anovan(y,group,param1,val1,param2,val2,...)`: 指定平方和类型计算方差分析，它可以是 1, 2 或 3，默认为 3。

`[p,table] = anovan(...)`: 返回单元数组表中的 `anova` 表。

`[p,table,stats] = anovan(...)`: 返回 `stats` 结构，用于多元检验。

【例 6-7】液体推进剂偏二甲肼具有中等以上的毒性。为了达到排放标准，拟采用空气催化氧化法处理废液，实验研究了影响空气催化氧化法效果的各种因素，测定了催化剂投加量、空气量、通气时间等因素对偏二甲肼去除率的影响，结果见表 6-7。请分析影响处理效果的因素。

表 6-7 空气催化氧化法影响因素（去除率）

试验序号	催化剂投加量/(g/L)	空气量/(L/h)	通气时间/h	偏二甲肼去除率/%
1	20	200	1	33
2	20	200	3	62
3	20	400	1	37
4	20	400	3	63
5	40	200	1	58
6	40	200	3	75
7	40	400	1	63
8	40	400	3	80

其实现的 MATLAB 程序代码如下:

```
>> x=[33 62 37 63 58 75 63 80];
group={'cat1','cat1','cat1','cat1','cat2','cat2','cat2','cat2';['air1','air1','air2','air2','air1','air1','air2','air2'];[
'time1','time2','time1','time2','time1','time2','time1','time2']};
anovan(x,group,2,3,{'cat','air','time'})'
```

运行程序, 输出如下:

```
ans =
    0.0236    0.1257    0.0215    0.3440    0.0903    0.5000
```

从分析结果图 6-7 可看出, 催化剂和氧化时间对处理效果有显著影响, 而空气量影响不显著, 所以通过优化催化剂投加量和氧化时间可明显改善处理效果。

Analysis of Variance					
Source	Sum Sq.	d. f.	Mean Sq.	F	Prob>F
cat	820.13	1	820.125	729	0.0236
air	28.13	1	28.125	25	0.1257
time	990.12	1	990.125	880.11	0.0215
cat*air	3.12	1	3.125	2.78	0.344
cat*time	55.12	1	55.125	49	0.0903
air*time	1.13	1	1.125	1	0.5
Error	1.13	1	1.125		
Total	1898.88	7			

Constrained (Type III) sums of squares.

图 6-7 方差分析表

利用 anovan 函数, 对前面的单因素试验的方差分析和双因素试验的方差分析同样都可以进行, 只是编程时需要指定 group 值。下面利用 anovan 函数编写一个可以进行单因素方差分析、双因素分析的函数 anovans。

其源代码如下:

```
function table=anovans(A, reps)
alpha=[0.05,0.01];
format short g;
if (nargin<2)||(~isnumeric(reps))||(floor(reps~=reps))||(reps<1))
    reps=0;
end
[m,n]=size(A);
if(reps>1)
    if mod(m,reps)
        error('矩阵的行数必须是单元行数的倍数');
    else
        m=reps;
    end
else
    m=1;
end
B=A;k=0;A=[];
while length(B)>0
    k=k+1;
    A(:,k)=B(1:m,:);
```

```

A1(:,k)=k*ones(m,n);
A2(:,k)=repmat(1:n,m,1);
B(1:m,:)=[];
end
X=A(:);A1=A1(:);A2=A2(:);
grp{1,1}=A2;
if reps>=1
    grp{1,1}=A1;
    grp{2,1}=A2;
end
if reps>=2
    mld='interaction';
else
    mld='linear';
end
[p,table]=anovan(X,grp,mld,[],[],'off');
table(:,4:6)=table(:,5:7);
table(1,:)={'方差来源','偏差平方和','自由度','方差','F 值','Fa','显著性'};
table((end-1):end,1)={'误差','总和'};
N=length(p);
table{2,1}='行间';
if N>=2
    table{2,1}='行间';
    table{3,1}='列间';
end
if N>=3
    table{4,1}='交互';
end
for k=1:N
    lft=finv(1-max(alpha),table{k+1,3},table{end-1,3});
    rgt=finv(1-min(alpha),table{k+1,3},table{end-1,3});
    table{k+1,6}=[num2str(lft),' ',num2str(rgt)];
    if p(k)<min(alpha)
        table{k+1,7}='高度显著';
    elseif (p(k)>=min(alpha)&p(k)<max(alpha))
        table{k+1,7}='显著';
    else
        table{k+1,7}='不显著';
    end
end
end

```

【例 6-8】对例 6-5，用 anovans 函数进行方差分析。

其实现的 MATLAB 程序代码如下：

```

>> clear all;
a=[31    37    35    38;34    36    37    39;...
    33    34    38    42;35    35    34    43;...
    33    39    33    38;34    36    40    37];
table=anovans(a,2)

```

运行程序，输出如下：

```
table =
Columns 1 through 5
'方差来源'    '偏差平方和'    '自由度'    '方差'    'F 值'
'行间'        [    3.0833]    [     2]    [1.5417]    [0.38144]
'列间'        [   114.46]    [     3]    [38.153]    [ 9.4399]
'交互'        [   36.917]    [     6]    [6.1528]    [ 1.5223]
'误差'        [    48.5]    [    12]    [4.0417]    [      ]
'总和'        [   202.96]    [    23]    [      ]    [      ]

Columns 6 through 7
'Fa'          '显著性'
'3.8853;6.9266' '不显著'
'3.4903;5.9525' '高度显著'
'2.9961;4.8206' '不显著'
           []      []
           []      []
```

6.4 多元方差分析

单因素多元方差分析检验某变量是否受到其他一个或多个变量的影响。利用该分析过程可以分析因素之间的主效应，也可以分析因素之间的交互效应。

在 MATLAB 中的统计工具箱中提供了 `multcompare` 函数实现多元方差的分析。其调用格式如下：

```
c = multcompare(stats)
c = multcompare(stats,param1,val1,param2,val2,...)
[c,m] = multcompare(...)
[c,m,h] = multcompare(...)
[c,m,h,gnames] = multcompare(...)
```

【例 6-9】多元方差分析示例。

其实现的 MATLAB 程序代码如下：

```
>> strength = [82 86 79 83 84 85 86 87 74 82 ...
               78 75 76 77 79 79 77 78 82 79];
alloy = {'st','st','st','st','st','st','st','st',...
         'al1','al1','al1','al1','al1','al1',...
         'al2','al2','al2','al2','al2','al2'};
[p,a,s] = anova1(strength,alloy);
>> [c,m,h,nms] = multcompare(s);
[nms num2cell(c)]
>> title('点击你所需测试的组');
```

运行程序，输出结果如下，效果如图 6-8 及图 6-9 所示。

```
ans =
'st'    [1]    [2]    [ 3.6064]    [ 7]    [10.3936]
'al1'   [1]    [3]    [ 1.6064]    [ 5]    [ 8.3936]
'al2'   [2]    [3]    [-5.6280]   [-2]    [ 1.6280]
```

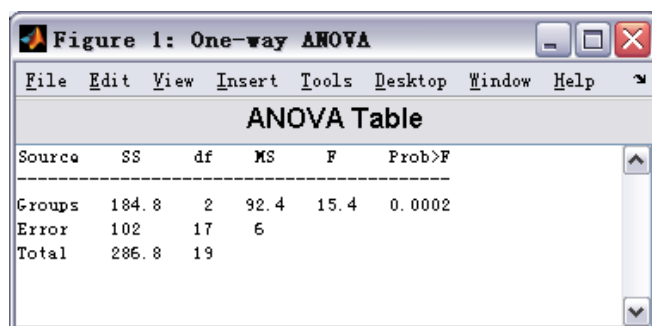


图 6-8 方差分析表

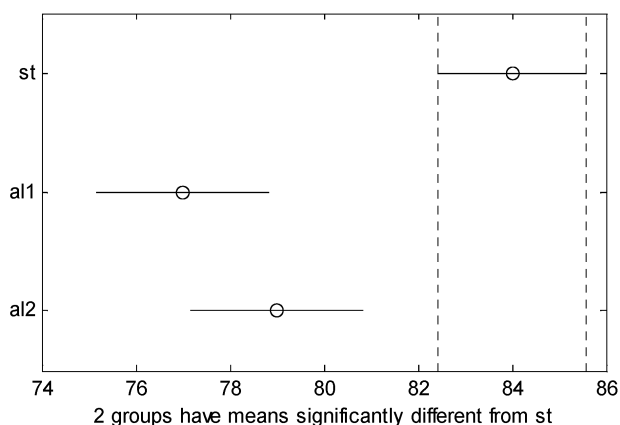


图 6-9 多元方差分析图

6.5 进一步讨论方差分析

6.5.1 把方差表输出到 Excel 中

用函数 `anova1`、`anova2c`、`anova3` 生成的方差分析表，有时可能需要插入到文件中，这可以先把方差分析表输出到一个 Excel 表格文件中，再根据需要插入到文件中。

其实现的 MATLAB 程序代码如下：

```
function outtable(table)
[m,n]=size(table);
[fil,pth]=uiputfile('方差分析表','保存方差分析表');
fid=fopen([pth,'\fil','.xls'],'w');
if fid>1
    fprintf(fid,'\t\t\t');
    fprintf(fid,'%s','方差分析表');
    fprintf(fid,'\n');
    for kh=1:m%行 4
        for kl=1:n%列 7
            temp=table{kh,kl};
            if length(temp)>0
                if isnumeric(temp)
```



```

        temp=num2str(temp);
    end
    fprintf(fid,'%s',temp);
    fprintf(fid,'\t');
end
end
fprintf(fid,'\n');
end
t=fopen(fid);

```

对于用 MATLAB 工具箱中的函数 `anova1`、`anova2`、`anova` 等生成的方差分析表，也可以用 `outtable` 函数输出到 Excel 表格中。

【例 6-10】对例 6-5 中的数据，用先 `anova2c` 函数生成方差分析表，再用函数 `outtable` 输出到 Excel 表格中。

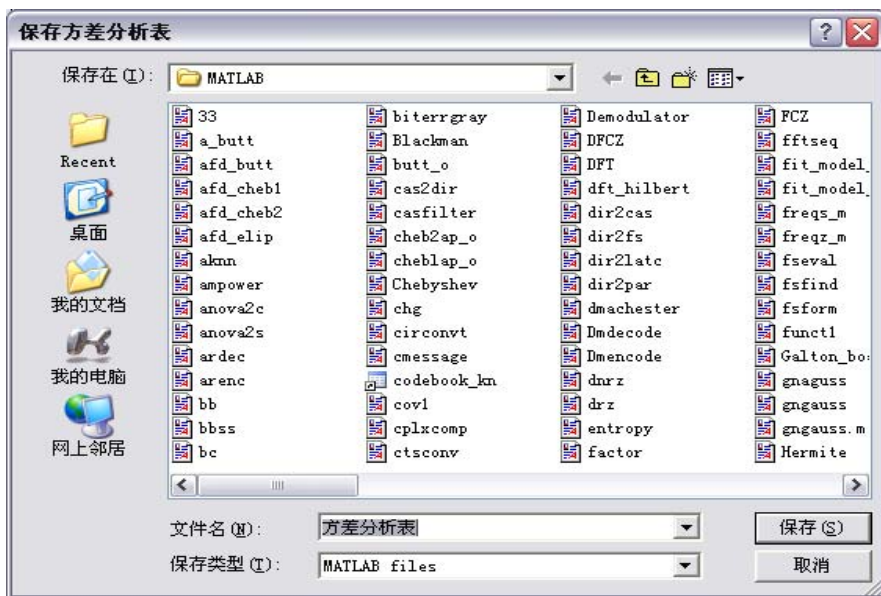


图 6-10 保存生成的方差分析表

其实现的 MATLAB 程序代码如下：

```

>> clear all;
a=[31 37 35 38;34 36 37 39;...
    33 34 38 42;35 35 34 43;...
    33 39 33 38;34 36 40 37];
table=anova2c(a,2);
outtable(table)

```

运行程序，输出结果如下，效果如图 6-10 所示。

```

status =
    0

```

默认的文件保存在当前目录下，文件名是“方差分析表”。单击“保存”按钮，就会在“方差分析”文件夹中生成名为“方差分析表”的 Excel 文件。其中的数据就是由“`table=anova2c(a,2);`”生成的方差分析表，如图 6-11 所示。

	1	2	3	4	5	6
1 '方差来源'	'偏差平方和'	'自由度'	'方差'	'F值'	'Fa'	
2 '因素a(行...'	3.0833	2	1.5417	0.3814	3.8853;8	
3 '因素b(列...'	114.4583	3	38.1528	9.4399	3.4903;5	
4 '交互作用'	36.9167	6	6.1528	1.5223	2.9961;4	
5 '误差'	48.5000	12	4.0417	[]	[]	
6 '总和'	202.9583	23	[]	[]	[]	

图 6-11 Execl 表中的方差分析表

6.5.2 方差表在图形窗口中的显示

在默认情况下,用函数 `anova1`、`anova2`、`anovan` 都会把方差分析表显示在一个图形窗口中,也可以通过编程,使得用函数 `anova1`、`anova2c`、`anova2s` 和 `anovans` 等生成的方差分析表,显示在图形窗口中。

其实现的 MATLAB 程序代码如下:

```
function displaytable(table,name)
if nargin<2
    name='方差分析表';
end
figw=0.9;
figh=0.4;
figl=0.5-figw/2;
figd=0.50-figh/2;
figNumber=figure(...
    'Units','normalized',...
    'Name','方差分析表',...
    'Position',[figl,figd,figw,figh],...
    'Resize','on',...
    'MenuBar','none',...
    'NumberTitle','off');
axl=0.05;
axd=0.05;
axw=0.9;
axh=0.8;
hh=axes(...
    'Units','normalized',...
    'Position',[axl,axd,axw,axh],...
    'NextPlot','add',...
    'Box','on',...
    'XTick',[],'YTick',[],'ZTick',[],...
    'tag','haxes');
```

```

[m,n]=size(table);
row=linspace(axh,axd,m);
col=linspace(axl,axw,n);
rowh=row(2)-row(1);
colw=col(2)-col(1);
begg=0.06;
text(0.5,1.05,name,'fontsize',18,'fontweight','bold',...
    'HorizontalAlignment','center');
for kh=1:m
    for kl=1:n
        temp=table{kh,kl};
        if length(temp)>0
            if isnumeric(temp)
                temp=num2str(temp);
            end
            text(axl/2+col(kl),-rowh/2+row(kh),temp, ...
                'HorizontalAlignment','center');
        end
    end
end
x1=[col(2:end);col(2:end)]-colw/3;
y1=[0;1];
x2=[0;1];
y2=[row(1:end-1);row(1:end-1)];
plot(x1,y1,'k-',x2,y2,'k-')

```

这一函数有两个输入参数，第一个输入参数 `table` 是函数 `anovas`、`anova2s`、`avno2c` 生成的方差分析表的单元数组；第二个输入参数 `name` 可选，用于标注方差分析表上的名称，默认值是“方差分析表”。

用 `displaytable` 函数，还可以显示由 MATLAB 中函数 `anova1`、`anova2`、`anovan` 等生成的方差分析表。

【例 6-11】 对例 6-5 中的数据，用 `anova2c` 函数生成方差分析表，再用函数 `displaytable` 在图形窗口中显示出来。

其实现的 MATLAB 程序代码如下：

```

>> clear all;
a=[31    34    35    39
    34    36    37    39
    33    34    38    42
    35    35    34    43
    33    39    33    38
    34    36    40    37];
table=anova2c(a,2);
displaytable(table,'双因素等重复试验方差分析表')

```

运行程序，生成效果如图 6-12 所示。

双因素等重复试验方差分析表						
方差来源	偏差平方和	自由度	方差	F值	Fa	显著性
因素a(行间)	5.0833	2	2.5417	0.61616	8.8853;6.9268	不显著
因素b(列间)	123.125	3	41.0417	9.9495	8.4903;5.9528	高度显著
交互作用	34.25	6	5.7083	1.3838	2.9961;4.8206	显著
误差	49.5	12	4.125			
总和	211.9583	23				

图 6-12 双因素等重复试验方差分析表

第 7 章 估计及假设检验

7.1 参数的点估计

设总体 X 的分布中含有未知参数 θ , 从总体 X 中抽取样本 X_1, X_2, \dots, X_n 构造某个统计量 $\hat{\theta}(X_1, X_2, \dots, X_n)$ 作为参数 θ 的估计, 则称 $\hat{\theta}(X_1, X_2, \dots, X_n)$ 为参数 θ 的点估计量。在不致混淆的情况下, 统称估计量和估计值为估计, 并都简记为 $\hat{\theta}$ 。由于统计量是样本的函数, 因此对于不同的样本值, θ 的估计一般是不相同的。

当总体 X 的分布中含有 m 个未知参数 $\theta_1, \theta_2, \dots, \theta_m$ 时, 需要构造 m 个统计量。 $\hat{\theta}_1 = \hat{\theta}_1(X_1, X_2, \dots, X_n)$, $\hat{\theta}_2 = \hat{\theta}_2(X_1, X_2, \dots, X_n)$, \dots , $\hat{\theta}_m = \hat{\theta}_m(X_1, X_2, \dots, X_n)$ 分别作为 $\theta_1, \theta_2, \dots, \theta_m$ 的点估计量。

下面介绍两种求未知参数的点估计量 (或点估计值) 的方法。

7.1.1 矩估计法

设总体 X 的分布中含有未知参数 $\theta_1, \theta_2, \dots, \theta_m$ 。假定总体 X 的 $1 \sim m$ 阶原点矩都存在, 则有

$$\mu_k = \mu_k(\theta_1, \theta_2, \dots, \theta_m) = E(X^k) \quad (k=1, 2, \dots, m)$$

取样本的 k 阶原点矩 A_k 作为总体 k 阶原点矩 μ_k 的估计量, 即

$$\hat{\mu}_k = \frac{1}{n} \sum_{i=1}^n X_i^k \quad (k=1, 2, \dots, m)$$

由此得到方程组

$$\begin{aligned}\mu_1(\theta_1, \theta_2, \dots, \theta_m) &= \hat{\mu}_1 \\ \mu_2(\theta_1, \theta_2, \dots, \theta_m) &= \hat{\mu}_2 \\ &\vdots \\ \mu_m(\theta_1, \theta_2, \dots, \theta_m) &= \hat{\mu}_m\end{aligned}$$

求这个方程组的解, 得

$$\hat{\theta}_1 = \hat{\theta}_1(X_1, X_2, \dots, X_n), \quad \hat{\theta}_2 = \hat{\theta}_2(X_1, X_2, \dots, X_n), \quad \dots, \quad \hat{\theta}_m = \hat{\theta}_m(X_1, X_2, \dots, X_n)$$

就分别是参数 $\theta_1, \theta_2, \dots, \theta_m$ 的矩估计量。

【例 7-1】设总体 X 的均值 μ 及方差 σ^2 都存在, 且有 $\sigma^2 > 0$, 但 μ 、 σ^2 均未知, 又设 X_1, X_2, \dots, X_n 是来自 X 的样本, 试求 μ 、 σ^2 的矩估计值。

因为总体 X 的分布中有两个未知参数, 所以应考虑一、二阶原点矩, 有

$$\begin{aligned}\mu_1 &= E(X) = \mu \\ \mu_2 &= E(X^2) = D(X) + [E(X)]^2 = \sigma^2 + \mu^2\end{aligned}$$

于是, 按矩估计法得方程组

$$\begin{cases} \mu = \frac{1}{n} \sum_{i=1}^n X_i \\ \sigma^2 + \mu^2 = \frac{1}{n} \sum_{i=1}^n X_i^2 \end{cases}$$

解得 μ 及 σ^2 的矩估计量为

$$\begin{cases} \hat{\mu} = \frac{1}{n} \sum_{i=1}^n X_i = \bar{X} \\ \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n X_i^2 - \bar{X}^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2 \end{cases}$$

而 μ 及 σ^2 的矩估计值为

$$\begin{aligned} \hat{\mu} &= \frac{1}{n} \sum_{i=1}^n x_i = \bar{x} \\ \hat{\sigma}^2 &= \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \end{aligned}$$

所得结果表明, 总体均值与方差的矩估计量的表达式不因不同的总体分布而异。

例如, $X \sim N(\mu, \sigma^2)$, μ 、 σ^2 未知, 即得 μ 、 σ^2 的矩估计量为

$$\hat{\mu} = \bar{X}, \quad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2$$

矩估计法的优点是直观、简便。正如在例 7-1 中指出的, 对总体的均值与方差进行估计时, 并不一定要知道总体服从什么分布, 但是, 矩估计法对于那些原点不存在的总体是不适用的。总体矩不存在, 则矩法失效。

7.1.2 极大似然估计法

首先举例说明极大似然估计法的数学原理。

【例 7-2】设有甲、乙两个布袋, 甲袋中有 99 个白球和 1 个黑球, 乙袋中有 1 个白球和 99 个黑球。由于某种原因已不能识别哪一个甲袋, 哪一个是乙袋。你能否用统计的方法识别出来?

下面对这一问题进行数学描述与分析。

不妨设变量 X 表示袋中的白球数, 则 $X \sim \begin{pmatrix} 1 & 99 \\ p & 1-p \end{pmatrix}$, p 是未知的分布参数, 其取值依赖于

变量 X 代表的是甲袋中的白球数还是乙袋中的白球数。显然, 变量 X 代表的是甲袋中的白球数与 $p=99/100$ 是等价的, 变量 X 的代表是乙袋中的白球数与 $p=1/100$ 是等价的。

通过抽样 (任取一袋, 从该袋中任取一球, 观察其颜色) 的方法来确定 $p=99/100$ 还是 $p=1/100$ 。

设事件 A 表示 “取出的一袋为甲袋”, 事件 B 表示 “从袋子中取出的是白球”, 则

$$P(A)=0.5, \quad P(B|A)=99/100, \quad P(B|\bar{A})=1/100$$

假定取出的是白球。在已知取出的是白球的条件下, 判断该球来自甲袋还是乙袋的问题, 可由贝叶斯公式, 通过比较概率 $P(B|A)$ 和 $P(\bar{A}|B)$ 的大小来作出判断。由于在一次试验中大概

率事件容易发生, 因此, 若 $P(A|B) > P(\bar{A}|B)$, 则该球来自甲袋; 若 $P(A|B) < P(\bar{A}|B)$, 则该球来自乙袋。

因为

$$P(A|B) = \frac{P(AB)}{P(B)} = \frac{P(A)P(B|A)}{P(A)P(B|A) + P(\bar{A})P(B|\bar{A})}$$

$$P(\bar{A}|B) = \frac{P(\bar{A}B)}{P(B)} = \frac{P(\bar{A})P(B|\bar{A})}{P(A)P(B|A) + P(\bar{A})P(B|\bar{A})}$$

这两个式子的分母相同, 分子中 $P(A) = P(\bar{A})$, 故其大小取决于 $P(B|A)$ 和 $P(B|\bar{A})$ 的大小, 而 $P(B|A)$ 和 $P(B|\bar{A})$ 的取值恰好等于变量 X 的分布参数 p 的两个可能的取值。这说明参数的取值同逆概率 $P(B|A)$ 和 $P(B|\bar{A})$ 之间的大小是相互决定的, 即 $p=99/100$ 等价于 $P(A|B) > P(\bar{A}|B)$, $p=1/100$ 等价于 $P(A|B) < P(\bar{A}|B)$ 。

通过计算可知, $P(A|B) > P(\bar{A}|B)$, 因此 $p=99/100$, 即现在取出的这一袋是甲袋。

概括这里的思想方法, 就可以得到极大似然估计法的数学原理——大率原理: 大率事件在一次试验中容易发生。或者说, 在一次试验中已经发生的事件具有较大的概率, 而变量的分布参数有助于关于该变量的大率事件的发生。

接下来讨论参数的极大似然估计的方法。

设 $X_1, X_2, \dots, X_n \sim X$, 并记变量 X 的概率分布律或概率密度函数为 $p(x; \theta_1, \theta_2, \dots, \theta_k)$, 其中 $\theta_1, \theta_2, \dots, \theta_k$ 是变量 X 的 k 个未知参数。

又设对样本 (X_1, X_2, \dots, X_n) 进行一次观测得到样本值 (x_1, x_2, \dots, x_n) , 这相当于 n 个相互独立的事件 $\{X_1 = x_1\}, \{X_2 = x_2\}, \dots, \{X_n = x_n\}$ 在一次试验中同时发生, 即事件 $\{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n\}$ 应该有较大的概率值。

1) X 是离散变量的情形

根据前述极大似然估计法的数学原理, 可令

$$P\{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n\} = \prod_{i=1}^n P\{X_i = x_i\} = \prod_{i=1}^n P\{x_i; \theta_1, \theta_2, \dots, \theta_k\}$$

达到最大值, 此时对应的参数值 $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$ 即为参数真值 $\theta_1, \theta_2, \dots, \theta_k$ 的估计值。

2) X 是连续变量的情形

对连续变量考虑概率 $P\{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n\}$ 是没有意义的。因此, 考虑随机点 (X_1, X_2, \dots, X_n) 落入以点 (x_1, x_2, \dots, x_n) 为顶点、 $\Delta x_1, \Delta x_2, \dots, \Delta x_n$ 为边长的 n 维矩形区域 G 内的概率, 这个概率近似等于

$$P\{(X_1, X_2, \dots, X_n) \in G\} = \prod_{i=1}^n P\{x_i; \theta_1, \theta_2, \dots, \theta_k\} \cdot \prod_{i=1}^n \Delta x_i$$

同理, 可令这个概率达到最大值, 此时对应的参数值 $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$ 即为参数真值 $\theta_1, \theta_2, \dots, \theta_k$ 的估计值。

注意到 $\Delta x_i (i=1, 2, \dots, n)$ 与 $\theta_1, \theta_2, \dots, \theta_k$ 无关, 使 $\prod_{i=1}^n p(x_i; \theta_1, \theta_2, \dots, \theta_k) \prod_{i=1}^n \Delta x_i$ 达到最大值的点

$(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k)$ 与使 $\prod_{i=1}^n P\{x_i; \theta_1, \theta_2, \dots, \theta_k\}$ 达到最大值的点相同, 而后者在表达形式上连续型变量与

离散变量是一致的, 因此给出下面的定义。

定义 7-1 称样本 x_1, x_2, \dots, x_n 的联合概率函数 (概率分布律或概率密度函数)

$$L(\theta) = L(x_1, x_2, \dots, x_n; \theta) = \prod_{i=1}^n p\{x_i; \theta\}$$

为参数 $\theta = (\theta_1, \theta_2, \dots, \theta_k)$ 的似然函数。

设 Θ 为参数 θ 所有可能的取值范围, 称为参数空间。若存在统计量 $\hat{\theta} \in \Theta$, 使得

$$L(x_1, x_2, \dots, x_n; \hat{\theta}) = \max_{\theta \in \Theta} L(x_1, x_2, \dots, x_n; \theta)$$

则称 $\hat{\theta}$ 为参数 θ 的极大似然估计量 (Maximum Likelihood Estimate, MLE)。

求似然函数 $L(\theta)$ 的极大值一般情况下要先求其驻点, 涉及导数运算。由于似然函数 $L(\theta)$ 的数学表达式往往是积与幂的结构, 其导数运算会比较冗繁, 不方便求驻点, 而对数函数 $\ln x$ 是 x 的单调增函数, 因此对数似然函数 $\ln L(\theta)$ 与似然函数 $L(\theta)$ 在同一点处取得最大值。又对数能够将积运算转化为和运算, 将幂运算转化为积运算, 从而使似然函数 $L(\theta)$ 的数学表达式线性化, 方便导数与求驻点运算。于是, 通常情况下应当先将似然函数 $L(\theta)$ 转化为对数似然函数 $\ln L(\theta)$, 然后再求驻点。

【例 7-3】求事件 A 发生的概率 p 的极大似然估计。

令 $X = \begin{cases} 1, & \omega \in A \\ 0, & \omega \notin A \end{cases}$, 其中 $\omega \in A$ 表示事件 A 发生, 则 X 的概率函数为

$$p(x; p) = p^x (1-p)^{1-x} \quad (x=0,1)$$

故参数 p 的似然函数为

$$L(p) = \prod_{i=1}^n p^{x_i} (1-p)^{1-x_i} = p^{\sum_{i=1}^n x_i} (1-p)^{\sum_{i=1}^n (1-x_i)}$$

对数似然函数为

$$\ln L(p) = \left(\sum_{i=1}^n x_i \right) \ln p + \left(n - \sum_{i=1}^n x_i \right) \ln(1-p)$$

对 p 求导数, 令导数为 0, 就有

$$\frac{d \ln L(p)}{dp} = \frac{1}{p} \left(\sum_{i=1}^n x_i \right) - \frac{1}{1-p} \left(n - \sum_{i=1}^n x_i \right) = 0$$

解得 $\ln L(p)$ 的驻点为

$$p = \frac{1}{n} \sum_{i=1}^n x_i$$

又在驻点处有

$$\frac{\partial^2 \ln L(p)}{\partial p^2} = \frac{-n}{p(1-p)} < 0$$

所以, 驻点即为极大值点, 即 p 的极大似然估计为 $\hat{p} = \bar{x}$ 。

表 7-1 列出了参数估计函数, 其返回值为数据向量 x 的参数最大似然估计值, 以及置信度为 $(1-a) \times 100\%$ 的置信区间。 a 的默认值为 0.05, 即置信度为 95%。

表 7-1 参数估计函数

函数名	调用格式	函数说明
binofit	phat = binofit(x,n)	二项分布的概率最大似然估计
	[phat,pci] = binofit(x,n)	置信度为 95% 的参数估计和置信区间
	[phat,pci] = binofit(x,n,alpha)	返回水平 α 的参数估计和置信区间
poissfit	lambdshat = poissfit(data)	泊松分布的参数最大似然估计
	[lambdahat,lambdaci] = poissfit(data)	置信度为 95% 的参数估计和置信区间
	[lambdahat,lambdaci] = poissfit(data,alpha)	返回水平 α 的参数估计和置信区间
normfit	[muhat,sigmahat] = normfit(data)	正态分布的最大似然估计, 置信度为 95%
	[muhat,sigmahat,muci,sigmaci] = normfit(data,alpha)	返回水平 α 的期望、方差值和置信区间
betafit	phat = betafit(data)	返回 β 分布参数 a 和 b 的最大似然估计
	[phat,pci] = betafit(data,alpha)	返回最大似然估计值和水平 α 的置信区间
unifit	[ahat,bhat] = unifit(data)	均匀分布参数的最大似然估计
	[ahat,bhat] = unifit(data)	置信度为 95% 的参数估计和置信区间
	[ahat,bhat,ACI,BCI] = unifit(data,alpha)	返回水平 α 的参数估计和置信区间
expfit	muhat = expfit(data)	指数分布参数的最大似然估计
	[muhat,muci] = expfit(data)	置信度为 95% 的参数估计和置信区间
	[muhat,muci] = expfit(data,alpha)	返回水平 α 的参数估计和置信区间
gamfit	phat = gamfit(data)	r 分布参数的最大似然估计
	[phat,pci] = gamfit(data)	置信度为 95% 的参数估计和置信区间
	[phat,pci] = gamfit(data,alpha)	返回最大似然估计值和水平 α 的置信区间
wblfit	parmhat = wblfit(data)	韦伯分布参数的最大似然估计
	[parmhat,parmc] = wblfit(data)	置信度为 95% 的参数估计和置信区间
	[parmhat,parmc] = wblfit(data,alpha)	返回水平 α 的参数估计及其区间估计
mle	phat=mle('dist', data)	分布函数名为 dist 的最大似然估计
	[phat, pci]=mle('dist', data)	置信度为 95% 的参数估计和置信区间
	[phat, pci]=mle('dist', data, alpha)	返回水平 α 的最大似然估计值和置信区间
	[phat, pci]=mle('dist', data, alpha, p1)	仅用于二项分布, p1 为试验总次数

【例 7-4】求泊松分布的极大似然估计值和置信区间。

其实现的 MATLAB 程序代码如下：

```
>> r = poissrnd(5,10,2);
[l,pci] = poissfit(r)
```

运行程序，输出泊松分布的极大似然估计值和置信区间如下：

```
l =
    6.3000    4.5000
lci =
    4.8411    3.2823
    8.0604    6.0214
```

【例 7-5】随机产生 100 个服从正态分布 $N(2,0.5^2)$ 的样本数据 X ，并用这些数据估计总体

$N(\mu, \sigma^2)$ 中的参数 μ 、 σ ，求出参数的最大似然估计值和置信水平为 99% 的置信区间。

分析：随机产生的 100 个数据可视为总体中抽出容量为 100 的样本，样本的观测值就是这具体的 100 个数据，可用命令 `normfit(X, alpha)` 求出参数 μ 、 σ 的估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
X=normrnd(2,0.5,100,1); %产生 100 个样本数据
[muhat,sigmahat,muci,sigmaci]=normfit(X,0.01)
```

运行程序输出结果如下：

```
muhat =      2.0240
sigmahat =    0.4343
muci =
    1.9099
    2.1380
sigmaci =
    0.3665
    0.5298
```

说明：参数 μ 、 σ 的估计最大似然值分别为 2.0240、0.4343，参数 μ 、 σ 的置信水平为 99% 的置信区间分别为 [1.9099, 2.1380]、[0.3665, 0.5298]。这一估计结果和总体 $N(\mu, \sigma^2)$ 中的参数真实数值 $\mu=2$ 、 $\sigma=0.5$ 是非常接近的。

可以概括出求极大似然估计值的一般步骤：

- (1) 明确变量的分布律和密度函数。
- (2) 写出似然函数 $L(\theta)$ 。
- (3) 求似然函数 $L(\theta)$ 的最大值点，得 $\hat{\theta}_{MLE}$ 。
- (4) 应用问题中，将样本数据代入 $\hat{\theta}_{MLE}$ 求出具体的估计值。

值得注意的是，求解对数似然方程组是在假定其可导并且导数变号的基础上，若不满足这一条件，需针对似然函数 $L(\theta_1, \theta_2, \dots, \theta_k)$ 的单调性，利用极大似然估计的基本原理直接进行 $L(\theta_1, \theta_2, \dots, \theta_k)$ 最大值问题的讨论。

极大似然估计量有一个简单而有用的性质：设 θ 的函数 $g = g(\theta)$ 是 Θ 上的实值函数，且有唯一反函数。如果 $\hat{\theta}$ 是 θ 的极大似然估计量，则 $g(\hat{\theta})$ 也是 $g(\theta)$ 的极大似然估计量。这个性质称为极大似然估计的不变性。根据这一性质可以使一些复杂结构的参数的极大似然估计问题简单化。

极大似然估计法是在变量分布类型已知的情况下使用的一种参数估计法。一般地，用极大似然法所得的估计的性质比用矩估计法所得的要好，故通常多用极大似然法。

【例 7-6】通常，引用常数的测定值服从均值为 μ 、标准差为 σ 的正态分布。某人在实验中使用金球测定引力常数，6 次测定观察值为：6.683, 6.681, 6.676, 6.678, 6.679, 6.672。试用极大似然估计法对未知参数 μ 和 σ 作出估计。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x=[6.683,6.681,6.676,6.678,6.679,6.672];
phat=mle(x,'distribution','norm','alpha',0.05)
```

运行程序输出如下：

```
phat =
```

6.6782 0.0035

即金球测定的 μ 的估计值为 6.6782, σ 的估计值为 0.0035。另外, 此例计算中 `mle` 函数的调用可以简化为 `p=mle(x)`。

7.1.3 估计量的评选标准

1. 无偏性

设 $\hat{\theta} = \hat{\theta}(X_1, X_2, \dots, X_n)$ 的数学期望等于 θ , 即

$$E(\hat{\theta}) = \theta$$

则称 $\hat{\theta}$ 是参数 θ 的无偏估计量; 若样本观测值为 x_1, x_2, \dots, x_n , 则称 $\hat{\theta}(x_1, x_2, \dots, x_n)$ 为参数 θ 的无偏估计值。

在科学技术中 $E(\hat{\theta}) - \theta$ 称为以 $\hat{\theta}$ 作为 θ 的估计的系统误差, 无偏估计的实际意义就是无系统误差。

无偏性是对估计量的一个最重要、最常见的要求, 它的实际意义在于, 当这个估计量经常使用时, 在多次重复的平均意义下, 给出了接近于真值 θ 的估计, 在此应当指出, 同一个参数 θ 的无偏估计量不是唯一的。例如, $E(X_i) = \mu$, 这表明任一样本的每一分量 X_i ($i=1, 2, \dots, n$) 都是总体均值 μ 的无偏估计量, 在参数 θ 的许多无偏估计中, 当然是以对 θ 的平均偏差较小者为好, 即较好的估计量应当有尽可能小的方差。因此, 便有了第二个评选标准。

2. 有效性及最优性

一个未知参数 θ 的估计量 $\hat{\theta}$ 仅有无偏性是不够的。因为一方面, 无偏性仅反映估计量在参数真值周围波动, 而没有反映出“集中”的程度; 另一方面, 一个参数的无偏估计量可能不止一个, 对于数学期望 μ , 样本均值 \bar{X} 是它的无偏估计量, 样本的第一个观测值 X_1 也是它的无偏估计量 (因 $E(X_1) = \mu$), 那么哪个更好呢? 仅有无偏性一个标准是不能确定的。一个自然的想法是进一步比较它们的方差, 方差越小, 表示 $\hat{\theta}$ 越集中在 θ 的附近, 从这个意义上讲方差越小的无偏估计量越好。

设 $\hat{\theta}_1, \hat{\theta}_2$ 都是 θ 的无偏估计量, 若 $D(\hat{\theta}_1) < D(\hat{\theta}_2)$, 则称 $\hat{\theta}_1$ 比 $\hat{\theta}_2$ 有效。

【例 7-7】试比较总体期望 μ 的两个无偏估计量 $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$ 及 $\hat{\alpha} = X_1$ 的有效性。

设总体方差为 σ^2 , 则

$$D(\bar{X}) = \frac{1}{n} \sigma^2, \quad D(\hat{\alpha}) = D(X_1) = \sigma^2$$

显然 $D(\bar{X}) < D(\hat{\alpha})$, 故 \bar{X} 较 $\hat{\alpha}$ 有效。

可以证明, 无偏估计 $\hat{\theta}$ 的方差 $D(\hat{\theta})$ 有一个非零的下界, 即最小方差 $D_0(\hat{\theta})$, 有

$$D(\hat{\theta}) \geq \frac{1}{nE\left(\left(\frac{\partial \ln f(X; \theta)}{\partial \theta}\right)^2\right)} = D_0(\hat{\theta})$$

其中 $f(X; \theta)$ 为总体分布的概率密度函数。如果 $\ln f(X; \theta)$ 存在关于 θ 的二阶偏导数, 则可证明

$$D(\hat{\theta}) \geq \frac{1}{-nE\left[\left(\frac{\partial^2 \ln f(X;\theta)}{\partial \theta^2}\right)^2\right]} = D_0(\hat{\theta})$$

若 $\hat{\theta}$ 满足 $E(\hat{\theta}) = \theta$, $D(\hat{\theta}) = D_0(\theta)$, 则称 $\hat{\theta}$ 为 θ 的方差一致最小无偏估计量, 写为 UMVUE 估计。

可以证明 \bar{X} 和 S^2 是总体均值 μ 和方差 σ^2 的 UMVUE 估计, \bar{X} 和 S^2 使用率很高, 一是由于 μ 和 σ^2 是很重要且常用的总体参数, 二是 μ 和 σ^2 有很好的统计性质。

3. 一致性

无偏性准则和均方误差准则是在样本容量 n 固定的情形下讨论估计量优劣的。设变量 $X \sim F(x)$, $\hat{F}_n(x)$ 为样本的经验分布函数, 由 $\Gamma_{\text{ЛИБЕЛКО}}$ 定理

$$P\{\lim_{n \rightarrow \infty} \sup_{-\infty \rightarrow x \rightarrow +\infty} |\hat{F}_n(x) - F(x)| = 0\} = 1$$

当样本容量 n 趋向于无穷时, 样本的经验分布函数以概率 1 一致收敛于变量的分布函数。也就是说, 当样本容量 n 趋向于无穷时, 样本中包含的关于变量分布的信息不断增加, 以致充分到可以将变量分布刻画到任意精确的程度。因此, 有理由要求, 一个“好的”估计量, 当样本容量 n 趋向于无穷时, 在一定的数学意义下收敛于被估参数。

设 $\hat{\theta}(X_1, X_2, \dots, X_n)$ 为参数 θ 的估计量, 若对任意的 $\varepsilon > 0$, 有

$$\lim_{n \rightarrow \infty} P\{|\hat{\theta} - \theta| \geq \varepsilon\} = 0$$

而且这对 θ 的一切可能取的值都成立, 则称 $\hat{\theta}$ 是参数 θ 的一个相合估计。

相合性准则是对一个估计量最基本的要求。它说明, 随着样本容量的增大, 一个“好的”估计量 $\hat{\theta}$ 应该越来越靠近参数 θ 的真值, 使绝对偏差 $|\hat{\theta} - \theta|$ 较大的概率越来越小。如果一个估计量没有相合性, 那么, 不论样本取多大, 也不可能把未知参数估计到预定的精度。这种估计量显然是不可取的。

下面, 不加证明地列举出关于相合估计的几个重要结论。

(1) 相合估计具有不变性。即当 $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k$ 分别是 $\theta_1, \theta_2, \dots, \theta_k$ 的相合估计时, 若 $g(\theta_1, \theta_2, \dots, \theta_k)$ 为连续函数, 则 $g(\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_k)$ 是 $g(\theta_1, \theta_2, \dots, \theta_k)$ 的相合估计。

(2) 样本 k 的阶原点矩 $A_k = \frac{1}{n} \sum_{i=1}^n X_i^k$ 是变量 X 的 k 阶原点矩 $E(X^k)$ 的相合估计, 故样本均值 \bar{X} 是变量均值 μ 的相合估计。

(3) 样本的二阶中心矩 $B_2 = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2$ 是变量 X 的方差 σ^2 的相合估计。

(4) 样本方差 $S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$ 是变量方差 σ^2 的相合估计, 样本标准差

$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2}$ 是变量标准差 σ 的相合估计。

(5) 事件发生的频率是其概率的相合估计。

(6) 极大似然估计量往往具有相合性。



7.2 区间估计

7.2.1 区间估计的基本概念

参数的点估计, 即怎样根据样本求得未知参数的点估计值。参数 θ 的点估计值只是 θ 的一个近似值。点估计的优点是简单明确; 缺点是没有提供一个精度的概念。对于一个未知量人们在使用时, 常不以得到近似值为满足; 还需要估计误差, 也就是要知道近似值的精确度。类似地, 对于未知参数 θ , 除了求出它的近似值外, 还希望估计出一个范围, 并希望知道这个范围包含参数 θ 真值的可信度。这样的范围通常以区间的形式给出, 同时还给出此区间包含参数 θ 真值的可信程度。这种形式的估计称为区间估计, 这样的区间即所谓置信区间。

若用 $\hat{\theta}(X_1, X_2, \dots, X_n)$ 做出未知参数 θ 的估计量, 其误差小于某个正数 ε 的概率为 $1-\alpha$ ($0 < \alpha < 1$), 即

$$P\{|\hat{\theta} - \theta| < \varepsilon\} = 1 - \alpha \quad (7-1)$$

也即

$$P\{\hat{\theta} - \varepsilon < \theta < \hat{\theta} + \varepsilon\} = 1 - \alpha \quad (7-2)$$

上式表明, 随机区间 $(\hat{\theta} - \varepsilon, \hat{\theta} + \varepsilon)$ 包含 θ 的真值的概率为 $1-\alpha$, 通常称区间 $(\hat{\theta} - \varepsilon, \hat{\theta} + \varepsilon)$ 为置信区间, 称 $1-\alpha$ 为置信水平 (或置信度)。置信区间表示估计结果的精确性, 而置信水平表示估计结果的可靠性。

设总体 X 的分布中含有未知参数 θ , 若对于给定值 α ($0 < \alpha < 1$), 存在两个统计量 $\hat{\theta}_1 = \hat{\theta}_1(X_1, X_2, \dots, X_n)$ 与 $\hat{\theta}_2 = \hat{\theta}_2(X_1, X_2, \dots, X_n)$, 使得

$$P\{\hat{\theta}_1 < \theta < \hat{\theta}_2\} = 1 - \alpha \quad (7-3)$$

则称随机区间 $(\hat{\theta}_1, \hat{\theta}_2)$ 为参数 θ 的置信水平为 $1-\alpha$ 的置信区间, $\hat{\theta}_1$ 称为置信下限, $\hat{\theta}_2$ 称为置信上限。

式 (7-3) 的含义是若反复抽样多次 (每次样本容量都是 n), 那么每个样本值都确定一个区间 $(\hat{\theta}_1, \hat{\theta}_2)$, 于是每个这样的区间, 要么包含真值 θ , 要么不包含真值 θ 。按式 (7-3), 在这众多的区间中, 包含真值 θ 的约占 $100(1-\alpha)\%$, 不包含真值 θ 的约占 $100 \times \alpha\%$ 。例如, 若 $\alpha=0.01$, 反复抽样 1000 次, 则得到的 1000 个区间中不包含真值 θ 的约占 10 个。

那么, α 取多大才好呢? 显然, α 越小, $1-\alpha$ 越大, 这说明区间 $(\hat{\theta}_1, \hat{\theta}_2)$ 包含 θ 的可靠性越好。是否 α 越小越好呢? 不是, 因为 α 越小, 导致区间 $(\hat{\theta}_1, \hat{\theta}_2)$ 的长度就会越大, 如果长度过大, 区间估计就失去了它的意义。是否区间越小越好呢? 也不是, 因为区间置信水平 $1-\alpha$ 就会变小, 即 $(\hat{\theta}_1, \hat{\theta}_2)$ 包含真值 θ 的可靠性就会缩小, 区间估计也失去了它的意义。所以正确的提法是: 先保证可靠度, 在保证可靠性的基础上尽量提高精确度。也就是说, 在给定较大的置信水平 $1-\alpha$ 下 (通常取 $1-\alpha=0.95$), 使 $(\hat{\theta}_1, \hat{\theta}_2)$ 长度最小的区间估计, 才是最好的区间估计。

综上所述, 区间估计就是在给定 α 值下, 由样本 (X_1, X_2, \dots, X_n) 去求两个统计量 $\hat{\theta}_1$ 和 $\hat{\theta}_2$ 的问题。

在 MATLAB 的统计学工具箱中提供了多变量线性回归参数估计与置信区间估计的函数 regress, 可以求出其所需的估计结果。其调用格式如下:



```
b = regress(y,X)
[b,bint] = regress(y,X)
[b,bint,r] = regress(y,X)
[b,bint,r,rint] = regress(y,X)
[b,bint,r,rint,stats] = regress(y,X)
[...] = regress(y,X,alpha)
```

【例 7-8】假设线性回归方程为 $y = x_1 - 1.232x_2 + 2.23x_3 + 2x_4 + 4x_5 + 3.792x_6$ ，试生成 120 组随机输入值 x_i ，计算出输出向量 y 。以这些信息为已知，观察是否能由最小二乘方法得出待定系数的估计值，并得出置信区间。

其实现的 MATLAB 程序代码如下：

```
>> a=[1 -1.232 2.23 2 4 3.792]';
X=randn(120,6);
y=X*a;
a1=inv(X'*X)*X'*y      %inv 为求最小二乘解
a1 =
    1.0000
   -1.2320
    2.2300
    2.0000
    4.0000
    3.7920
```

可见，因为输出值完全由精确计算得出，所以线性回归参数估计的误差是及其微小的，可以忽略。用 `regress` 函数还可以计算出 99% 置信度的置信区间。

其实现的 MATLAB 程序代码和输出结果如下：

```
>> [a,aint]=regress(y,X,0.01)
a =
    1.0000
   -1.2320
    2.2300
    2.0000
    4.0000
    3.7920
aint =
    1.0000    1.0000
   -1.2320   -1.2320
    2.2300    2.2300
    2.0000    2.0000
    4.0000    4.0000
    3.7920    3.7920
```

假设观测的输出数据已被噪声污染，则可以给输出样本叠加上 $N(0, 0.5)$ 区间的正态分布噪声，这时可以用下面语句进行线性回归分析，得出待定系数向量的估计参数及置信区间，用 `errorbar` 函数还可以用图形绘制参数估计的置信区间。

```
>> yhat=y+sqrt(0.5)*randn(120,1);
[a,aint]=regress(yhat,X,0.01)
```

```
errorbar(1:6,a,aint(:,1)-a,aint(:,2)-a)
```

运行程序，输出结果如下，效果如图 7-1 所示。

```
a =
    0.9345
   -1.1014
    2.1483
    2.0271
    3.9228
    3.8744
aint =
    0.7434    1.1257
   -1.2796   -0.9232
    1.9469    2.3497
    1.8344    2.2198
    3.7313    4.1143
    3.6974    4.0514
```

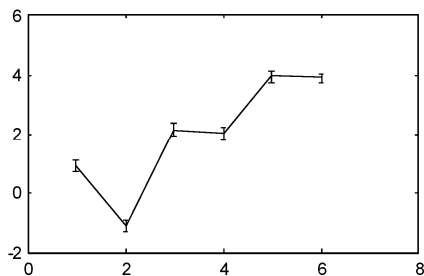


图 7-1 噪声 $\sigma^2=0.5$ 的参数估计的置信区间效果图

假设方差为 0.1，则可以得出新噪声下参数估计的结果。

```
>> yhat=y+sqrt(0.1)*randn(120,1);
[a,aint]=regress(yhat,X,0.01)
errorbar(1:6,a,aint(:,1)-a,aint(:,2)-a);
```

运行程序，效果如图 7-2 所示。显然估计出的参数更精确。

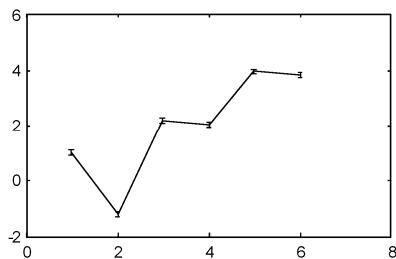


图 7-2 噪声 $\sigma^2=0.1$ 的参数估计的置信区间效果图

7.2.2 高斯—牛顿法的非线性最小二乘数据拟合

使用 `nlinfit` 函数求解高斯—牛顿法的非线性最小二乘数据拟合，其调用格式如下：

`beta = nlinfit(X,y,fun,beta0)`: 返回在 `fun` 中描述的非线性函数的系数。`fun` 为用户提供的形如 $\hat{y} = f(\beta, x)$ 的函数，该函数返回已给初始参数估计值 β 和自变量 x 的 y 的预测值 \hat{y} 。

`[beta,r,J,COVB,mse] = nlinfit(X,y,fun,beta0)`: 除了实现上面命令的功能外, 返回的 `beta` 为拟合系数, `r` 为残差, `J` 为 Jacobian 矩阵, `COVB` 为评估的协方差矩阵, `mse` 为误差的方差, `beta0` 为初始预测值。

`[...] = nlinfit(X,y,fun,beta0,options)`: 指定控制参数后返回值。参数 `options` 包括 `MaxIter`、`TolFun`、`TolX`、`Display`、`DerivStep` 等。

当 `X` 为矩阵时, 则 `X` 的每一列为自变量的取值, `y` 是一个相应的列向量。如果 `fun` 中使用了 `@`, 则表示函数的句柄。

【例 7-9】高斯—牛顿法的非线性最小二乘数据拟合示例。

其实现的 MATLAB 程序代码如下:

```
>> load reaction
beta = nlinfit(reactants,rate,@hougen,beta)
```

运行程序, 输出如下:

```
beta =
    1.2526
    0.0628
    0.0400
    0.1124
    1.1914
```

7.2.3 非线性模型的参数置信区间

使用 `nlparci` 函数求解非线性模型的参数估计的置信区间。其调用格式如下:

`ci = nlparci(beta,resid,'jacobian',J)`: 返回置信度为 95% 的置信区间, `beta` 为非线性最小二乘法估计的参数值, `resid` 为残差, `J` 为 Jacobian 矩阵。`nlparci` 函数可以用 `nlinfitt` 函数的输出作为其输入。

`ci = nlparci(beta,resid,'covar',sigma)`: 返回置信度为 95% 的置信区间, `sigma` 为协方差矩阵系数。

`ci = nlparci(...,'alpha',alpha)`: 返回 $100(1-\alpha)\%$ 的置信区间。

【例 7-10】非线性模型的参数置信区间示例。

其实现的 MATLAB 程序代码和输出结果如下:

```
>> load reaction
[beta,resid,J,Sigma] = ...
    nlinfit(reactants,rate,@hougen,beta)
beta =
    1.2526
    0.0628
    0.0400
    0.1124
    1.1914
resid =
    0.1321
   -0.1642
   -0.0909
    0.0310
    0.1142
    0.0498
```



```

-0.0262
0.3115
-0.0292
0.1096
0.0716
-0.1501
-0.3026
J =
6.8739 -90.6525 -57.8634 -1.9288 0.1614
3.4454 -48.5350 -13.6239 -1.7030 0.3034
5.3563 -41.2094 -26.3039 -10.5216 1.5095
1.6950 0.1091 0.0186 0.0278 1.7913
2.2967 -35.5653 -6.0537 -0.7567 0.2023
11.8669 -89.5648 -170.1730 -8.9565 0.4400
4.4973 -14.4261 -11.5409 -9.3769 2.5744
4.1831 -41.7891 -16.8935 -5.7794 1.0082
11.8285 -51.3718 -154.1151 -27.7408 1.5001
9.1514 -25.5946 -76.7838 -30.7135 2.5790
3.3373 0.0900 0.0720 0.1079 3.5269
9.3663 -102.0600 -107.4317 -3.5811 0.2200
4.7512 -24.4628 -16.3086 -10.3001 2.1141
Sigma =
0.7517 0.0377 0.0267 0.0648 -0.7244
0.0377 0.0019 0.0013 0.0033 -0.0363
0.0267 0.0013 0.0010 0.0023 -0.0258
0.0648 0.0033 0.0023 0.0056 -0.0622
-0.7244 -0.0363 -0.0258 -0.0622 0.7001
>> ci = nlparci(beta,resid,'jacobian',J)
ci =
-0.7467 3.2519
-0.0377 0.1632
-0.0312 0.1113
-0.0609 0.2857
-0.7381 3.1208
>> errorbar(1:5,beta,ci(:,1)-beta,ci(:,2)-beta)

```

运行程序，效果如图 7-3 所示。

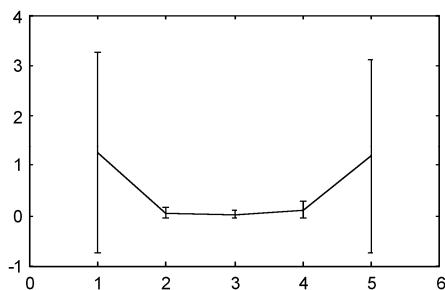


图 7-3 参数估计与置信区间效果

7.2.4 非线性最小二乘预测置信区间

使用 `nlpredci` 函数求解非线性模型置信区间预测。其调用格式如下：

`[ypred,delta] = nlpredci(modelfun,x,beta,resid,'jacobian',J)`: 返回预测值 `ypred`, `fun` 与前面相同, `beta` 为给出的适当参数, `resid` 为残差, `J` 为 Jacobian 矩阵, `x` 为非线性函数中的独立变量的矩阵值。返回的 `delta` 为非线性最小二乘法估计的置信区间长度的一半。当 `resid` 长度超过 `beta` 的长度, 并且 `J` 的列满秩时, 置信区间的计算才是有效的, `[ypred-delta, ypred+delta]` 为置信度, 是 95% 的不同步置信区间。

`[ypred,delta] = nlpredci(modelfun,x,beta,resid,'covar',sigma)`: 返回值同上类似, `sigma` 为协方差矩阵系数。

`[...] = nlpredci(...,param1,val1,param2,val2,...)`: 根据参数及其值返回, 这些参数包括 '`alpha`'、'`mse`'、'`predopt`' 和 '`simopt`'。

【例 7-11】 非线性最小二乘预测置信区间示例。

其实现的 MATLAB 程序代码如下：

```
>> load reaction;
[beta,resid,J,Sigma] = nlinfit(reactants,rate,@hougen,beta);
newX = reactants(1:2,:);
[ypred, delta] = nlpredci(@hougen,newX,beta,resid,'Covar',Sigma)
```

运行程序, 输出如下：

```
ypred =
    8.4179
    3.9542
delta =
    0.2805
    0.2474
```

7.2.5 非线性拟合预测的交互图形

使用 `nlintool` 函数求解非线性拟合并显示交互图形。其调用格式如下：

`nlintool(X,y,fun,beta0)`: 返回数据(`x`, `y`)的非线性曲线的预测图形, 它用两条红色曲线预测全局置信区间。`beta0` 为参数的初始预测值, 默认值为 0.05, 即置信度为 95%。

`nlintool(X,y,fun,beta0,alpha)`: 除了实现上面命令的功能外, 还将置信度设置为 $(1-\alpha) \times 100\%$ 。

`nlintool(X,y,fun,beta0,alpha,'xname','yname')`: 给 `x` 和 `y` 的变量分别赋予变量名 `xname` 和 `yname`。

【例 7-12】 非线性拟合的预测的交互图形示例。

其实现的 MATLAB 程序代码如下：

```
>> load reaction
nlintool(reactants,rate,@hougen,beta,0.01,xn,yn)
```

运行程序, 效果如图 7-4 所示。

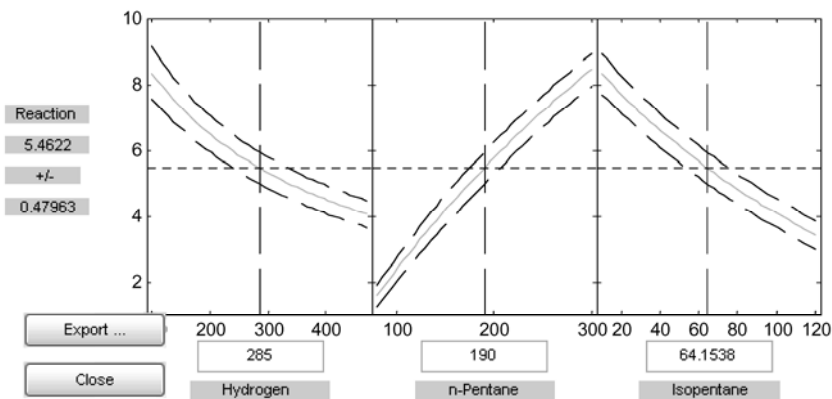


图 7-4 非线性拟合并显示交互图形效果

7.3 假设检验

统计推断的另一类重要问题是假设检验问题。先对总体的某个未知参数或总体的分布形式作某种假设，然后由所抽取的样本提供的信息，构造合适的统计量，对所提出的假设进行检验，以做出统计判断。是接受假设还是拒绝假设，这类统计推断问题称为假设检验问题。

7.3.1 假设检验的概念及步骤

先假设总体具有某种统计特征（如具有某种参数或遵从某种分布），然后再检验这个假设是否可信，这种方法称为统计假设检验方法。统计假设检验在统计学中是有重要地位的。例如，有人提出这样的假设，某灯泡工厂生产的某种型号的灯泡平均寿命在 3000h 以上，如何检验这个假设是否正确。该方法的确切检验方法，即将所有灯泡使用到烧坏为止显然是没有意义的，而应该采用统计假设检验的方法对该假设进行检验。下面将通过例子来演示系统假设检验实现的步骤。

【例 7-13】已知某产品的平均强度为 $\mu_0=9.94\text{kg}$ ，现在改变制作方法，并从新产品中随意抽取 200 件，算得它们的平均强度为 $\bar{x}=9.73\text{kg}$ ，标准差 $s=1.62\text{kg}$ ，问制作方法的改变对强度有无显著影响。

解决这样的问题需要采用统计假设检验，其步骤如下：

(1) 引入两个命题如下：

$$\begin{cases} H_0: \mu = \mu_0, \text{即无显著改变} \\ H_1: \text{拒绝 } H_0 \text{ 假设} \end{cases}$$

(2) 选取统计量

$$u = \frac{\sqrt{n}(\bar{x} - \mu_0)}{s} \quad (7-4)$$

该统计量满足标准正态分布 $N(0,1)$ 。对本例来说，可以计算出统计量如下：

```
>> clear all;
n=200;nu0=9.94;
xbar=9.73;s=1.62;
```

```
u=sqrt(n)*(nu0-xbar)/s
u =
    1.8332
```

(3) 给出显著性水平，由于统计检验毕竟不是确切性检验，所以无论接受还是拒绝该假设都有可能出错。引入 α 的意义是判定出现“取伪”错误的概率。由于研究的是随机问题，当然不可能令 $\alpha=0$ 。一般经常取 $\alpha=5\%$ 或 $\alpha=2\%$ ，用语言表示即为“可以有 95%（或 98%）的把握接受或拒绝该假设”。

(4) 有了 α 值，则可以用逆正态分布函数求出 $K_{\alpha/2}$ 的值，使得

$$\int_{-K_{\alpha/2}}^{K_{\alpha/2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx < 1 - \alpha \quad (7-5)$$

该步骤可以用 MATLAB 求解，可以得出不同 α 值下的 $K_{\alpha/2}$ 值，如下所示。其中，第一列为 α 的值，第二列为相应的 $K_{\alpha/2}$ 值。

```
>> alpha=[0.01:0.01:0.05,0.07,0.09];
K=norminv(1-alpha/2,0,1);
[alpha',K']
ans =
    0.0100    2.5758
    0.0200    2.3263
    0.0300    2.1701
    0.0400    2.0537
    0.0500    1.9600
    0.0700    1.8119
    0.0900    1.6954
```

(5) 由式 (7-4) 计算出统计量 u 的值，若 $|u| < K_{\alpha/2}$ ，则不拒绝 H_0 假设，否则拒绝该假设。该步骤实现的 MATLAB 程序代码如下：

```
>> abs(u)<K
ans =
     1     1     1     1     1     0     0
```

从比较表达式结果可见，在 $\alpha=0.01 \sim 0.05$ 的显著性水平下均可以接受假设 H_0 ，认为改进方法后无显著变化，但增大 α 的值，如这里的 $\alpha=0.07, 0.09$ ，则应该拒绝该假设，然而这样结论的可靠性将较低，因为出现错误的概率比较大。

7.3.2 总体参数的假设检验

在实际应用中，常涉及有关总体参数的假设检验问题。下面对总体参数的假设检验展开介绍。

设 θ 为总体参数， θ_0 为一个已知数。常见的关于此参数的检验问题有 3 类：

第 1 类为

$$H_0: \theta = \theta_0 \leftrightarrow H_1: \theta \neq \theta_0 \quad (7-6)$$

第 2 类为

$$H_0: \theta \leq \theta_0 \leftrightarrow H_1: \theta > \theta_0 \quad (7-7)$$

第3类为

$$H_0: \theta \geq \theta_0 \leftrightarrow H_1: \theta < \theta_0 \quad (7-8)$$

称假设检验问题式(7-6)为双边假设检验问题,或双边检验问题;称假设检验问题式(7-7)或式(7-8)为单边假设检验问题,或单边检验问题。

对于上面的单边或双边检验问题,应该基于 θ 的一个好的点估计 $\hat{\theta}$ 来构造检验统计量。特别地,当用 $T = \hat{\theta} - \theta_0$ 作为检验统计量时,对于双边检验,拒绝域应该取如下的形式:

$$\{x \in \mathbb{R} : |x| > x_0\} \quad (7-9)$$

此时尾概率为

$$P_{H_0}(|T| > |t_0|) = \sup_{\theta \leq \theta_0} P_{\theta}(T > t_0) \quad (7-10)$$

对于单边检验式(7-8),通常把拒绝域取成如下的形式:

$$\{x \in \mathbb{R} : x < x_0\} \quad (7-11)$$

此时尾概率为

$$P_{H_0}(T < t_0) = \sup_{\theta \geq \theta_0} P_{\theta}(T < t_0) \quad (7-12)$$

7.4 单正态假设检验

正态分布总体均值的假设检验有着极广泛的应用,其理论上的结果也比较完善,一般的统计软件都能直接处理这类假设检验问题。

7.4.1 单正态 U 检验

已知总体方差情况下的均值检验方法简称为“U 检验”。

假设总体变量 $X \sim N(\mu, \sigma_0^2)$, X_1, X_2, \dots, X_n 为 X 的重复观测样本,其中总体标准差 σ_0 为已知实数。感兴趣的假设检验问题分别为

$$H_0: \mu = \mu_0 \quad (7-13)$$

$$H_0: \mu \leq \mu_0 \quad (7-14)$$

$$H_0: \mu \geq \mu_0 \quad (7-15)$$

其中 μ_0 为已知实数。

此时样本均值 \bar{X} 为总体均值的一个好的点估计,因此可以用

$$Z = \frac{\sqrt{n}(\bar{X} - \mu_0)}{\sigma_0}$$

作为检验统计量。

假设已经获得的重复观测数据为 x_1, x_2, \dots, x_n , 可以计算

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k, \quad z_0 = \frac{\sqrt{n}(\bar{x} - \mu_0)}{\sigma_0}$$

对于双边检验问题式(7-13),其尾概率等于

$$P_{(\mu_0, \sigma_0)}(|Z| > |z_0|) \quad (7-16)$$

对于单边检验问题式 (7-14)，其尾概率等于

$$P_{H_0}(Z > z_0) = \sup_{\mu \leq \mu_0} P_{(\mu, \sigma_0)}(Z > z_0) \quad (7-17)$$

对于单边检验问题式 (7-15)，其尾概率等于

$$P_{H_0}(Z < z_0) = \sup_{\mu \geq \mu_0} P_{(\mu, \sigma_0)}(Z < z_0) \quad (7-18)$$

借助于这些尾概率的计算公式，可以解决前面提到的假设检验问题。

注意：

(1) 这里 $P_{(\mu, \sigma_0)}$ 表示当 $X \sim N(\mu, \sigma_0^2)$ 检验统计量 Z 的概率分布。

(2) 当 $X \sim N(\mu, \sigma_0^2)$ 时，得 $\bar{X} \sim N\left(\mu, \frac{\sigma_0^2}{n}\right)$ ，这样就可以得到

$$P_{(\mu, \sigma_0)}(Z < z_0) = P_{(\mu, \sigma_0)}\left(\frac{\sqrt{n}(\bar{X} - \mu)}{\sigma_0} < \frac{\sqrt{n}(\bar{x} - \mu)}{\sigma_0}\right) = \Phi\left(\frac{\sqrt{n}(\bar{x} - \mu)}{\sigma_0}\right)$$

其中， Φ 为标准正态分布的分布函数。计算尾概率需要上面这个公式。

在 MATLAB 中，U 检验法使用 `ztest` 函数进行假设检验。其调用格式如下：

`h = ztest(x,m,sigma)`：返回显著性水平为 0.05 的假设检验，其中 x 为正态分布的样本， m 为均值 μ_0 ， σ 为标准差。

`h = ztest(...,alpha)`：将显著性水平改为 α 。

`h = ztest(...,alpha,tail)`：根据 `tail` 指定的备择假设进行假设检验。

`h = ztest(...,alpha,tail,dim)`：根据 x 指定的维数进行假设检验。

`h = ztest(...)`：返回 h 指定的维数进行假设检验。

`[h,p,ci,zval] = ztest(...)`：返回的 h 为假设检验， ci 为真正均值 μ 的 $(1-\alpha)$ 的置信区间， $zval$ 为统计量的值， ci 和 $zval$ 可缺省。 p 为观察值的概率，当 p 为小概率时，则对原假设提出质疑。

原假设

$$H_0: \mu = \mu_0 = m$$

当 `tail=0` 时，表示备择假设 $H_1: \mu \neq \mu_0 = m$ （默认，双边检验）。

当 `tail=1` 时，表示备择假设 $H_1: \mu > \mu_0 = m$ （单边检验）。

当 `tail=-1` 时，表示备择假设 $H_1: \mu < \mu_0 = m$ （单边检验）。

默认的 `tail` 值为 0。

当 `h=0` 时，表示在显著性水平 α 下，不能拒绝原假设。

当 `h=1` 时，表示在显著性水平 α 下，可以拒绝原假设。

【例 7-14】方差已知时的单个样本均值检验。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
%产生样本
```

```

%样本点数
N=1024;
%正态分布
x1=randn(1,N);
m1=mean(x1)
sigma1=1;
%假设检验
[h1,sig1,ci1]=ztest(x1,0,sigma1)
%Weibull 分布
x2=wblrnd(1,2,N,1);
m2=mean(x2)
sigma2=1;
%假设检验
[h2,sig2,ci2]=ztest(x2,0,sigma2)

```

运行程序，输出如下：

```

m1 =
    0.0045
h1 =
     0
sig1 =
    0.8865
ci1 =
   -0.0568    0.0657

```

即接受 x_1 的均值等于零。

```

m2 =
    0.8901
h2 =
     1
sig2 =
   1.9273e-178
ci2 =
    0.8288
    0.9513

```

即不接受 x_2 的均值等于零。

【例 7-15】某车间用一台包装机包装葡萄糖，每袋装的糖重一个随机变量，它服从正态分布。当机器正常时，其均值为 0.5kg，标准差为 0.015。某日开工后检验包装机是否正常，随机地抽取所包装 9 袋糖，称得净重 (kg) 为

0.498 0.505 0.512 0.489 0.521 0.543 0.481 0.52 0.511

问机器是否正常？

其实现的 MATLAB 程序代码如下：

```

>> clear all;
X=[0.498    0.505    0.512    0.489    0.521    0.543    0.481    0.52    0.511];
[h,p,ci,zval]=ztest(X,0.5,0.015,0.05,0)

```

运行程序，输出如下：

```

h =
     0

```

```
p =
    0.0754
ci =
    0.4991    0.5187
zval =
    1.7778
```

结果表明： $h=0$ 说明在水平 $\alpha=0.05$ 下，可以接受原假设，即认为包装机工作正常。

7.4.2 单正态 t 检验

未知总体方差情况下的均值检验为 t 检验。

假设总体变量 $X \sim N(\mu, \sigma^2)$ ，其中， σ 为未知实数， X_1, X_2, \dots, X_n 为 X 的重复观测样本。感兴趣的还是总体均值的双边检验问题式 (7-13)、单边检验问题式 (7-14) 和式 (7-15)。

此时 $Z = \frac{\sqrt{n}(\bar{X} - \mu_0)}{\sigma}$ 含有未知的总体标准差 σ ，不是检验统计量。为解决此问题，用样本标准差代替总体标准差，得到检验统计量

$$T = \frac{\sqrt{n}(\bar{X} - \mu_0)}{S}$$

此外 $S = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (X_k - \bar{X})^2}$ 。

注意：由于这里用样本标准差来近似总体标准差，这种近似会使得检验结果犯第一类错误的概率变大，因此，当已知总体标准差时应该基于检验统计量 Z 确定检验准则；当总体标准差未知时，才基于检验统计量 T 确定检验准则。

假设已经获得的重复观测数据 x_1, x_2, \dots, x_n ，可以计算

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k, \quad s_0 = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2}, \quad t_0 = \frac{\sqrt{n}(\bar{x} - \mu_0)}{s_0}$$

对于双边检验问题式 (7-13)，其尾概率等于

$$P_{(\mu, \sigma_0)}(|T| < |t_0|) \quad (7-19)$$

对于单边检验问题式 (7-14)，其尾概率等于

$$P_{H_0}(T > t_0) = \sup_{\mu \leq \mu_0} P_{(\mu, \sigma)}(T > t_0) \quad (7-20)$$

对于单边检验问题式 (7-154)，其尾概率等于

$$P_{H_0}(T < t_0) = \sup_{\mu \geq \mu_0} P_{(\mu, \sigma)}(T < t_0) \quad (7-21)$$

借助于这些尾概率的计算公式，可以解决前面提到的假设检验问题。

在 MATLAB 中提供了 `ttest` 函数来计算式 (7-19)、式 (7-20) 及式 (7-21) 中的尾概率。其调用格式如下：

$h = \text{ttest}(x)$ ：返回 μ 的显著性水平为 0.05 的假设检验，其中 x 为正态分布的样本，均值 μ 为 0。

$h = \text{ttest}(x, m)$ ：返回 μ 的显著性水平为 0.05 的假设检验，其中 x 为正态分布的样本，均值 μ

为 m 。

$h = ttest(x,y)$: 返回 x 和 y 差异的假设检验, x 、 y 的维数必须相同。

$h = ttest(...,\alpha)$: α 为给定显著性水平。

$h = ttest(...,\alpha,tail)$: 根据 $tail$ 指定的备择假设进行假设检验。

$h = ttest(...,\alpha,tail,dim)$: 根据 x 指定的维数进行假设检验。

$[h,p,ci,stats] = ttest(...)$: p 为观测值概率, 当 p 为小概率时则对原假设提出质疑, ci 为真正均值 μ 的 $(1-\alpha)$ 置信区间, ci 和 $stats$ 可缺省。 $stats$ 为统计构造的一些值, 包括如下:

(1) $tstat$ ——检验统计的值。

(2) df ——检验的次数。

(3) sd ——样本标准背离。

原假设

$$H_0: \mu = \mu_0 = m$$

当 $tail=0$ 时, 表示备择假设 $H_1: \mu \neq \mu_0 = m$ (默认, 双边检验)。

当 $tail=1$ 时, 表示备择假设 $H_1: \mu > \mu_0 = m$ (单边检验)。

当 $tail=-1$ 时, 表示备择假设 $H_1: \mu < \mu_0 = m$ (单边检验)。

默认的 $tail$ 值为 0。

当 $h=0$ 时, 表示在显著性水平 α 下, 不能拒绝原假设。

当 $h=1$ 时, 表示在显著性水平 α 下, 可以拒绝原假设。

【例 7-16】 方差未知时的单个样本均值检验。

其实现的 MATLAB 程序代码如下:

```
>> clear all;
%产生样本
%样本点数
N=1024;
%正态分布
x1=randn(1,N);
m1=mean(x1)
%假设检验
alpha0=0.05;
[h1,p1,ci1]=ttest(x1,0, alpha0)
%Weibull 分布
x2=x1+0.5;
m2=mean(x2)
%假设检验
[h2,p2,ci2]=ttest(x2,0, alpha0)
```

运行程序, 输出如下:

```
m1 =
    0.0265
h1 =
     0
p1 =
    0.3931
ci1 =
```

```
-0.0344    0.0874
```

即在显著性水平 $\alpha = 0.05$ 下接受 x_1 的均值等于零。

```
m2 =
    0.5265
h2 =
     1
p2 =
    4.1639e-057
ci2 =
    0.4656    0.5874
```

即在显著性水平 $\alpha=0.05$ 下拒绝接受 x_2 的均值等于零。

【例 7-17】某种电子元件的寿命 X （以 h 计）服从正态分布， μ 、 σ^2 均未知。现测得 16 只元件的寿命如下：

```
160 278 198 200 236 257 270 167 150 250 194 224 137
185 167 255
```

问是否有理由认为元件的平均寿命大于 220 (h)？

其实现的 MATLAB 程序代码如下：

```
>> clear all;
X=[160 278 198 200 236 257 270 167 150 250 194 224 137 185 167 255];
[h,p,ci]=ttest(X,220,0.005,1)
```

运行程序，输出如下：

```
h =
     0
p =
    0.8457
ci =
    174.4465      Inf          %均值 225 在该置信区间内
```

结果表明： $h=0$ 表示在水平 $\alpha=0.05$ 下应该接受原假设 H_0 ，即认为元件的平均寿命不大于 220h。

【例 7-18】已知数据

```
x=[2 3 4 5 7 8 11 14 15 16 18 19]
y=[106.42 108.2 109.58 110 109.93 110.49 110.59 110.6 110.9 110.76 111 111.2]
```

建立 y 与 x 之间的函数关系，并检验残差 r 是否服从均值为零的正态分布。

分析：通过作散点图，猜测曲线的参数表达式，求出最佳参数，得到 y 与 x 之间的函数关系，计算出残差，检验残差 e_i 是否服从均值为零的正态分布。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x=[2 3 4 5 7 8 11 14 15 16 18 19];
y=[106.42 108.2 109.58 110 109.93 110.49 110.59 110.6 110.9 110.76 111 111.2];
plot(x,y,'rp');          %作散点图
A=polyfit(x,y,1)          %线性最小二乘拟合
y1=polyval(A,x);
plot(x,y,'p',x,y1,'r')    % 绘制拟合直线
e1=y-polyval(A,x)         %计算出残差
```

```
[h1,sig,ci]=ttest(e1,0,0.05); %用 t 检验来验证残差是否服从正态分布
[h2,P,Jbstat,CV]=lillietest(e1,0.05) %正态分布拟合的检验
```

运行程序，输出如下：

```
A =
    0.1804   108.1387
e1 =
   -2.0794   -0.4798    0.7198    0.9595    0.5287    0.9083    0.4672
   -0.0640    0.0557   -0.2647   -0.3855   -0.3658
h2 =
     0
P =
    0.2039
Jbstat =
    0.1995
CV =
    0.2418
```

说明：x 与 y 的线性最小二乘拟合直线方程为 $y=0.1804x+108.1387$ ，不管是 t 检验还是 lillietest 检验，都接受残差 e_i 服从均值为零的正态分布的假设，但要注意 lillietest 检验给出的 $P=0.2039$ 很小，说明虽然通过检验，但不是很理想，这点从拟合的直线（图 7-5）也能直观看出。

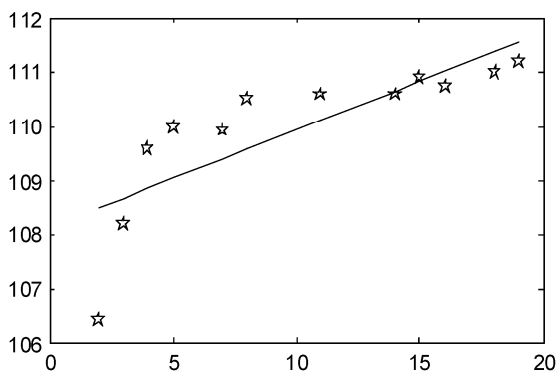


图 7-5 残差拟合直线

7.5 双正态假设检验

设有两个独立的正态总体 $X \sim N(\mu_1, \sigma_1^2)$ 、 $Y \sim N(\mu_2, \sigma_2^2)$ 、 X_1, X_2, \dots, X_{n_1} 与 Y_1, Y_2, \dots, Y_{n_2} 分别是 X 和 Y 的样本， \bar{X} 、 \bar{Y} 、 S_1^2 、 S_2^2 是相应的样本均值和样本方差。常见的关于均值的假设检验如下：

$H_0: \mu_1 = \mu_2$ ； $H_1: \mu_1 \neq \mu_2$ （称为双边检验， H_1 可略而不写）。

$H_0: \mu_1 = \mu_2$ ； $H_1: \mu_1 > \mu_2$ 或 $H_1: \mu_1 \leq \mu_2$ ； $H_1: \mu_1 > \mu_2$ （称为右边检验）。

$H_0: \mu_1 = \mu_2$ ； $H_1: \mu_1 < \mu_2$ 或 $H_1: \mu_1 \geq \mu_2$ ； $H_1: \mu_1 < \mu_2$ （称为左边检验）。

以 σ_1^2 、 σ_2^2 未知，但 $\sigma_1^2 = \sigma_2^2$ 为例，讨论检验假设 $H_0: \mu_1 = \mu_2$ ； $H_1: \mu_1 \neq \mu_2$ ，即得到统计量为

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_1 - \mu_2)}{S_w \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t(n_1 + n_2 - 2)$$

其中:

$$S_w^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}, \quad S_1^2 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (X_i - \bar{X})^2, \quad S_2^2 = \frac{1}{n_2 - 1} \sum_{i=1}^{n_2} (Y_i - \bar{Y})^2.$$

当 H_0 成立时, 统计量

$$T = \frac{(\bar{X} - \bar{Y}) - (\mu_1 - \mu_2)}{S_w \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t(n_1 + n_2 - 2)$$

于是, 对给定的显著性水平 α , 查 t 分布表, 取临界值 $t_{\frac{\alpha}{2}}(n_1 + n_2 - 2)$, 由

$$P\left(|T| \geq t_{\frac{\alpha}{2}}(n_1 + n_2 - 2)\right) = \alpha \quad \text{得 } H_0 \text{ 的拒绝域为}$$

$$|T_0| \geq t_{\frac{\alpha}{2}}(n_1 + n_2 - 2)$$

【例 7-19】在正常生产情况下, 印花棉布布幅的宽度服从正态分布 $N(1.4, 0.0048^2)$ 。某日选取该种棉布 5 匹, 测得布幅宽度为 1.32、1.55、1.36、1.40、1.44, 问该日印花棉布布幅宽度的标准差是否正常? (取 $\alpha=0.05$)

分析: 这是正态分布的方差检验问题。依题意, 令 $H_0: \sigma = 0.0048$; $H_1: \sigma \neq 0.0048$ 。检验统计量选样本方差 S^2 , 双侧检验, 检验准则为

$$P(S^2 \leq \delta_1 | \sigma = 0.0048) \leq \alpha/2 \quad \text{或} \quad P(S^2 \geq \delta_2 | \sigma = 0.0048) \leq \alpha/2$$

需对 S^2 进行变换以确定其概率分布。由抽样分布理论 $\chi^2 = \frac{(n-1)S^2}{\sigma_0^2} \sim \chi^2(n-1)$, 故在 H_0 成立

的条件下, $\chi^2 = 4S^2/0.0048^2 \sim \chi^2(4)$, 即

$$P\{\chi^2 \leq 4\delta_1/0.0048^2\} \leq \alpha/2 \quad \text{或} \quad P\{\chi^2 \geq 4\delta_2/0.0048^2\} \leq \alpha/2$$

由此可求出 H_0 拒绝域的临界值。

其实现的 MATLAB 程序代码如下:

```
>>clear all;
x=[1.32,1.55,1.36,1.40,1.44];
%求检验统计量的值
XVAR=var(x)
%求拒绝域的左侧临界值
DETA1=chi2inv(0.025,4)*0.0048^2/4
```

```
%求拒绝域的右侧临界值
DETA2=chi2inv(0.975,4)*0.0048^2/4
%求检验的 p 值
p=1-chi2cdf(4*XVAR/0.0048^2,4)
```

运行程序，输出如下：

```
XVAR =
    0.0078
DETA1 =
    2.7903e-006
DETA2 =
    6.4185e-005
p =
    0
```

由于检验统计量实测值 $S^2 = 0.0078 > \text{DETA2} = 0.000064185$ ，落入拒绝域，故否定原假设，即认为该日生产棉布布幅宽度的标准差不正常；检验的 p 值近似为零，表明作出这一结论的理由是充分的。

在 MATLAB 中， t 检验法采用 `ttest2` 函数比较两正态总体样本均值的假设检验，其调用格式如下：

`h = ttest2(x,y)`: 命令返回 μ 的显著性水平为 0.05 的两个正态分布样本均值的假设检验，其中 x 、 y 为正态分布的样本。

`h = ttest2(x,y,alpha)`: α 为给定显著性水平。

`h = ttest2(x,y,alpha,tail)`: 根据 `tail` 指定的备择假设进行假设检验。

`h = ttest2(x,y,alpha,tail,vartype)`: 根据 `vartype` 指定方差是否相等进行假设检验，`vartype` 可取 'equal' 或 'unequal'。

`h = ttest(x,y,alpha,tail,vartype,dim)`: 根据 x 、 y 指定的维数进行假设检验。

`[h,p,ci,stats] = ttest2(...)`: p 为观察值的概率，当 p 为小概率时则对原假设提出质疑， ci 为真正均值 μ 的 $(1-\alpha)$ 置信区间， ci 和 $stats$ 可缺省。 $stats$ 为统计构造的一些值，包括如下：

原假设

$$H_0: \mu = \mu_0 = m$$

当 `tail=0` 时，表示备择假设 $H_1: \mu \neq \mu_0 = m$ （默认，双边检验）。

当 `tail=1` 时，表示备择假设 $H_1: \mu > \mu_0 = m$ （单边检验）。

当 `tail=-1` 时，表示备择假设 $H_1: \mu < \mu_0 = m$ （单边检验）。

默认的 `tail` 值为 0。

当 `h=0` 时，表示在显著性水平 α 下，不能拒绝原假设。

当 `h=1` 时，表示在显著性水平 α 下，可以拒绝原假设。

【例 7-20】在同一只平炉上进行一项试验以确定改变操作方法的建议是否会增加钢的产率，每炼一炉钢时除操作方法外，其他条件都尽可能做到相同。先用标准方法炼一炉，然后用建议的新方法炼一炉，以后交替进行，各炼 10 炉，其产率分别如下：

标准方法: 78.2 72.5 73.3 77.8 79.5 75.5 76.5 76.9 75.2

新方法: 79.1 81.0 77.3 79.1 80.0 79.7 77.4 80.1 80.9

其实现的 MATLAB 程序代码如下：

```
>> clear all;
```

```
X=[78.2 72.5 73.3 77.8 79.5 75.5 76.5 76.9 75.2];
Y=[79.1 81.0 77.3 79.1 80.0 79.7 77.4 80.1 80.9];
[h,sig,ci]=ttest2(X,Y,0.05,-1)
```

运行程序，输出如下：

```
h =
    1
sig =
    0.0010           %说明两个总体均值相等的概率很小
ci =
   -Inf   -1.7055
```

结果表明： $h=1$ 表示在水平 $\alpha=0.05$ 下，应该拒绝原假设，即认为建议的新操作方法提高了产率，因此，比原方法好。

7.6 正态性检验

检验变量是否服从正态分布是统计应用中最常见也是最重要的问题。此类问题当然可以用 $K_{LJIMORPOB} - C_{MHPHOB}$ 检验法进行。但是，由于受样本容量因素的影响，有时检验效果可能不理想。因此，人们发现了一些专门的正态性检验方法，其检验效果一般比通用方法好。这里介绍三种常用的正态性检验方法。

1. 正态概率纸检验法

正态概率纸是一种现场统计常用的判断变量正态性的简单工具，使用它可以很快地判断变量是否服从正态分布，还能够粗略地估计出分布的数字特征。

首先介绍正态概率纸的构造原理。

设变量 X 的分布函数为 $F(x)$ ，需要检验

$$H_0: X \sim N(\mu, \sigma^2) \quad (-\infty < \mu < +\infty, \sigma^2 > 0)$$

在原假设 H_0 成立时， $\frac{X-\mu}{\sigma} = U \sim N(0,1)$ ，而且 $F(x)$ 可用标准正态分布 $N(0,1)$ 分布函数 $\Phi(x)$ 来表示：

$$F(x) = \Phi\left(\frac{X-\mu}{\sigma}\right) = \Phi(u)$$

其中：

$$u = \frac{1}{\sigma}(x - \mu)$$

在 xOu 直角坐标平面上，假定横轴 (x 轴) 与纵轴 (u 轴) 的单位长度相等，函数 $u = \frac{1}{\sigma}(x - \mu)$ 的图像是一条直线，过点 $(\mu, 0)$ ，斜率为 $\frac{1}{\sigma}$ 。

为使这条直线能够直观地解释变量的取值 x 与 $P\{X \leq x\}$ 之间的关系，进行如下坐标刻度更新：在直角坐标系 xOu 中，保持横轴上 x 的刻度不变，而把纵轴上 u 的刻度更新为 $y = 100\Phi(u)$ ，并规定 $100\Phi(-\infty) = 0$ ， $100\Phi(+\infty) = 100$ 。这样就将直角坐标 xOu 更新为直角坐标系 xOy 。由于 y 轴上的刻度 0 与 100 分别对应 u 轴上的 $-\infty$ 和 $+\infty$ ，因此 y 轴上无法标示出 0 与 100，一般 y 轴上的刻度标示限于 0.01~99.99。称以直角坐标系 xOy 为刻度体系的坐标纸为正态概率纸。

根据正态概率纸的构造原理可知,在 xOu 直角坐标系中 x 与 u 的关系,在 xOy 直角坐标系中就成为 x 与 $y=100P\{X\leq x\}(=100F(x)=100\Phi(u))$ 的关系;反之亦然。特别对于正态概率纸上的一条直线,若该直线能表示为 $u=\frac{1}{\sigma}(x-\mu)$,则 $100F(x)$ 与 x 的关系为

$$100F(x)=100\Phi(u)=100\Phi\left(\frac{x-\mu}{\sigma}\right)$$

即

$$F(x)=\Phi\left(\frac{x-\mu}{\sigma}\right)$$

也就是说, $F(x)$ 是一个正态分布的分布函数。

这表明,正态概率纸上斜率存在且大于零的全体直线所组成的集合与全体正态分布所组成的正态分布族之间存在一一对应关系。

【例 7-21】淮河流域(包括河南、安徽、江苏、山东)历史上经常发生洪水灾害,据统计 1949—1991 年流域成灾面积(单位:万亩^①)每年总计分别为:

338.4 4498.4 1635.7 2244.6 2011.8 6214.5 421.3 809.7 6234.8 5453.9 1412.4
312.0 2185.7 4079.2 10124.3 536.5 428.6 3978.2 249.6 2456.1 1055.7 1452.8
1563.5 789.5 1978.7 2478.6 736.2 515.6 379.5 37865.4 1025.5 2201.3 4407.1
2285.3 1124.5 1190 191.4 22275.3 2079 6321.1

试检验全流域的成灾面积是否服从正态分布。

其实现的 MATLAB 程序代码如下:

```
>> clear all;
X=[338.4 4498.4 1635.7 2244.6 2011.8 6214.5 421.3 809.7 6234.8 5453.9...
    1412.4 312.0 2185.7 4079.2 10124.3 536.5 428.6 3978.2 249.6 2456.1...
    1055.7 1452.8 1563.5 789.5 1978.7 2478.6 736.2 515.6 379.5 37865.4...
    1025.5 2201.3 4407.1 2285.3 1124.5 1190 191.4 22275.3 2079 6321.1];
normplot(X); %用概率纸检验数据是否服从正态分布
[h,P,Jbstat,CV]=jbtest(X,0.05) %正态分布拟合检验
title('绘制正态概率图');
ylabel('概率'); xlabel('数据');
```

运行程序,输出如下,效果如图 7-6 所示。

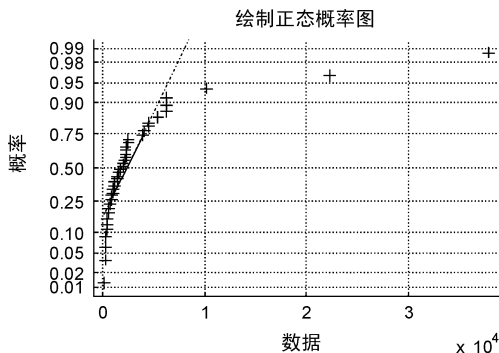


图 7-6 概率纸检验图

^①1 亩=666.7m²。

2. Jarque-Bera 检验

Jarque-Bera 检验是一种常用的、基于峰度与偏度联合检验的正态性检验方法。

设 $X_1, X_2, \dots, X_n \sim X$, X 的分布未知。需要检验

$$H_0: X \sim N(\mu, \sigma^2) \quad (-\infty < \mu < +\infty, \sigma^2 > 0)$$

令 $B_k = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^k$, Jarque 和 Bera 由样本峰度 $KU = \frac{B_4}{B_2^2}$ 和偏度 $SK = \frac{B_3}{B_2^{3/2}}$ 定义了如下的统计量:

$$J = \frac{n}{6} \left[SK^2 + \frac{(KU - 3)^2}{4} \right]$$

并证明了在 H_0 为真的条件下, J 渐近地服从自由度为 2 的 χ^2 分布。

由于正态分布的峰度 $KU=3$, 偏度 $SK=0$, 因此检验统计量 J 的观测值越大越对 H_0 不利。于是, 对于给定的显著性水平 α , 检验准则为 $P\{J > \chi_{1-\alpha}^2(2)\} \leq \alpha$ 。当检验统计量的实测值 $J > \chi_{1-\alpha}^2(2)$ 时, 则在显著性水平 α 下拒绝原假设 H_0 , 否则保留 H_0 。

由于检验依据是渐近分布, 因此该方法应在大样本条件下使用。

MATLAB 提供了 Jarque-Bera 检验法的检验函数 `jbtest`, 其调用格式如下:

`h = jbtest(x)`

`h = jbtest(x,alpha)`

`[h,p] = jbtest(...)`

`[h,p,jbstat] = jbtest(...)`

`[h,p,jbstat,critval] = jbtest(...)`

`[h,p,...] = jbtest(x,alpha,mctol)`

【例 7-22】 Jarque-Bera 检验法示例。

其实现的 MATLAB 程序代码如下:

```
>> clear all;
load carbig           %MATLAB 自带的数据库
[h,p] = jbtest(MPG)
h =
    1
p =
    0.0022
>> [h,p] = jbtest(log(MPG))
h =
    1
p =
    0.0078
>> [h,p] = jbtest(log(MPG),0.0075)
h =
    0
p =
    0.0078
```


3. Lilliefors 检验

Lilliefors 检验法是对 $K_{LJIMORPOB}$ 检验法的一种改进。

设 $X_1, X_2, \dots, X_n \sim X$, X 的分布未知。需要检验

$$H_0: X \sim N(\mu, \sigma^2) \quad (-\infty < \mu < +\infty, \sigma^2 > 0)$$

令 $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$, $S = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2}$, $Z_i = \frac{X_i - \bar{X}}{S}$ ($i=1, 2, \dots, n$), 则当 H_0 为真时, 标

准化样本 Z_1, Z_2, \dots, Z_n i.i.d. $\sim N(0, 1)$, 于是 $K_{LJIMORPOB}$ 统计量可修正为

$$D_n = \sup_{-\infty < x < +\infty} |S_n(x) - \Phi(x)|$$

式中, $S_n(x)$ 是标准化样本的经验分布函数。这就是 Lilliefors 检验的检验统计量。

其他如检验法则、检验步骤等与 $K_{LJIMORPOB}$ 检验法类似, 这里不再介绍。

由 Lilliefors 检验的检验统计量的构造特点可知, 该方法与 $K_{LJIMORPOB}$ 检验法的最大不同之处是检验不需要已知分布参数, 样本的标准化避免了在正态拟合优度检验之前对分布参数的估计, 因此该方法可在小样本条件下使用。

MATLAB 提供了 Lilliefors 检验法的检验函数 `lillietest`, 其调用格式如下:

`[h,p,stats,cv]=lillietest(x, alpha, tail)`

其输入、输出参数的意义同 `kstest`。

`h = lillietest(x)`

`h = lillietest(x,alpha)`

`h = lillietest(x,alpha,distr)`

`[h,p] = lillietest(...)`

`[h,p,kstat] = lillietest(...)`

`[h,p,kstat,critval] = lillietest(...)`

`[h,p,...] = lillietest(x,alpha,distr,mctol)`

【例 7-23】Lilliefors 检验法的检验示例。

其实现的 MATLAB 程序代码如下:

```
>> clear all;
load carbig.mat
[h,p] = lillietest(MPG)
Warning: P is less than the smallest tabulated value, returning 0.001.
> In lillietest at 170
h =
    1
p =
    1.0000e-003
>> [h,p] = lillietest(MPG,0.05,'norm',1e-4)
h =
    1
p =
    8.3333e-006
```

4. 其他的 Kolmogorov-Smirnov 检验

前面介绍的 Jarque-Bera 与 Lilliefors 假设检验算法和 MATLAB 函数只能用于检验某分布是否为正态分布，却不能用于其他分布的检验。Kolmogorov-Smirnov 检验是检验任意已知分布函数的一种有效的假设检验算法。MATLAB 的统计学工具箱中提供了 `kstest` 函数实现该算法。其调用格式如下：

```
h = kstest(x)
h = kstest(x,CDF)
h = kstest(x,CDF,alpha)
h = kstest(x,CDF,alpha,type)
[h,p,ksstat,cv] = kstest(...)
```

【例 7-24】Kolmogorov-Smirnov 检验示例。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x = -2:1:4
[h,p,k,c] = kstest(x,[],0.05,0)
xx = -3:1:5;
F = cdfplot(x);
hold on
G = plot(xx,normcdf(xx),'r-');
set(F,'LineWidth',2)
set(G,'LineWidth',2)
legend([F G],...
    '经验','标准正态',...
    '位置','NW')
title('经验 CDF');
```

运行程序，输出结果如下，效果如图 7-7 所示。

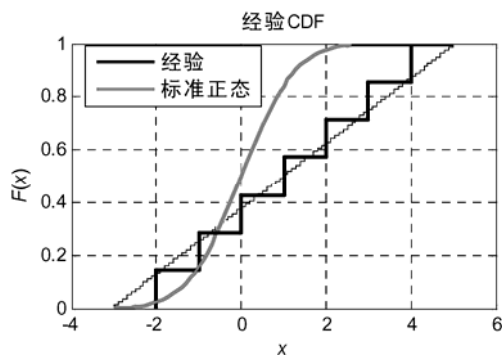


图 7-7 Kolmogorov-Smirnov 检验效果图

```
x =
    -2    -1     0     1     2     3     4
h =
     0
p =
    0.1359
k =
```

$$\begin{aligned} & 0.4128 \\ c = & \\ & 0.4834 \end{aligned}$$

7.7 总体参数检验

7.7.1 非正态总体样本的参数检验

前面讨论的假设检验都是针对正态总体, 而且对样本 n 没有任何条件限制, 也就是说, 无论 n 有多大, 其检验法都是有效的。但在实际应用中, 不时会遇到总体不服从正态分布的甚至不知道总体分布的情况。这时, 检验统计量及其分布便很难确定。在一般条件下, 中心极限定理对非正态总体成立, 因而常常可以借助于统计量的极限分布对总体参数做近似检验。这种检验要求样本容量 n 必须大, n 越大近似检验效果越好。要多大才好呢? 没有一个统一的标准, 因为这与所采用的统计量趋于它的极限分布的速度有关。实用上, 一般至少要求 $n \geq 30$, 最好 $n \geq 50$ 或 100。

对于非正态总体均值的假设检验, 以及两个非正态总体均值差异性的显著性检验, 在大样本的条件下, 都归结为 U 检验, 只是当方差 σ^2 已知时, 只要 $n \geq 50$ (至少 $n \geq 30$); 而当 σ^2 未知时, 通常用样本方差 S^2 去估计 σ^2 , 这时要求 $n \geq 100$, 才能保证检验的精度。

【例 7-25】一个市郊商业区林荫路的管理人员说, 每到周末, 停车场上汽车的平均停靠时间超过 90min, 随机抽查 100 辆周末到达该停车场的汽车, 算出平均停靠时间为 88min, 标准差为 30min。在 $\alpha=0.05$ 水平下, 检验管理员说法的真实性。

分析: 要判断的是汽车平均停靠时间是否超过 90min, 即要检验假设

$$H_0: \mu \geq 90; H_1: \mu < 90$$

因总体的分布类型和方差都未知, 但大样本 $n \geq 100$, 所以当 H_0 为真时, 统计量 $U = \frac{\sqrt{n}(\bar{X} - \mu_0)}{S}$ 近似服从正态分布 $N(0,1)$ 。

于是, 对给定的 $\alpha=0.05$, 查标准正态分布表, 取临界值 $u_\alpha=1.645$, 使

$$P\left(\frac{\bar{X} - \mu_0}{S/\sqrt{n}} \leq -u_\alpha\right) = \alpha$$

得 H_0 的拒绝域为 $\frac{\bar{X} - \mu_0}{S/\sqrt{n}} \leq -u_\alpha$ 。

计算得

$$U_0 = \frac{\bar{X} - \mu_0}{S/\sqrt{n}} = \frac{88 - 90}{30/\sqrt{100}} = -0.6667 > -1.645$$

故接受 H_0 , 即认为周末汽车平均停靠时间超过 90min。

【例 7-26】某厂产品的优质品率一直保持在 40%, 近期技监部门来厂抽查, 共抽查了 12 件产品, 其中优质品为 5 件, 在 0.05 显著性水平下能否认为其优质品率仍保持在 40%?

分析：设 X 表示检查一个产品时优质品的个数，则 $X \sim B(1, p)$ 。检验问题为

$$H_0: p = 0.4; \quad H_1: p \neq 0.4$$

这是一个双边检验问题。当 H_0 为真时，检验统计量 $T = \sum_{i=1}^n X_i \sim B(12, 0.4)$ ，拒绝域为 $T \leq c_1$ 或 $T \geq c_2$ ($c_1 < c_2$)。其中，临界值 c_1 是使 $P\{T \leq c_1\} \leq 0.025$ 成立的最大整数， c_2 是使 $P\{T \geq c_2\} \leq 0.025$ 成立的最小整数。

其实现的 MATLAB 程序代码如下：

```
>>clear all
%检验统计量的观测值
T=5;
%显著性水平
alpha=0.025;
%为确定拒绝域临界值计算 T 的累积概率
p=binocdf(0:12,12,0.4);
%求拒绝域临界值
for byk=1:7
if p(byk)<alpha&p(byk+1)>=alpha
c1=byk-1;
end
if (1-p(byk+6))>alpha&(1-p(byk+7))<=alpha
c2=byk+7;
end
end
%检验决策，h=1(0)拒绝（接受）原假设
if T<=c1|T>=c2
h=1
else
h=0
end
%输出拒绝域临界值
c=[c1,c2]
```

运行程序，输出如下：

```
h =
    0
c =
    1     9
```

上述计算表明，当 $\alpha=0.05$ 时，由于 $P\{T \leq 1\} < 0.025$ 而 $P\{T \leq 2\} > 0.025$ ，故拒绝域左侧临界值 $c_1=1$ ；又 $P\{T \geq 8\} > 0.025$ 而 $P\{T \geq 9\} < 0.025$ ，故拒绝域右侧临界值 $c_2=9$ 。于是， H_0 的拒绝域为 $T \leq 1$ 或 $T \geq 9$ 。检验统计量的观测值 $T = 5$ 未落入拒绝域，因而在 0.05 显著性水平下认为该厂优质品率无明显变化。

7.7.2 总体分布的 χ^2 拟合检验

设总体 X 的分布函数为具有明确表达式的 $F(x)$ （例如它可以属于正态分布、指数分布、泊松分布、二项分布等）。把随机变量 X 的值域 R 划分成 k 个互不相容的区间

$A_1 = (a_0, a_1], A_2 = (a_1, a_2], \dots, A_k = (a_{k-1}, a_k]$, 每个小区间的长度不一定相同。

设 x_1, x_2, \dots, x_n 是容量为 n 的样本的一组观测值, n_i 为样本观测值落入区间 A_i 内的频数,

$\sum_{i=1}^k n_i = n$, 则在 n 次试验中事件 A_i 出现的频率 $f(A_i) = \frac{n_i}{n}$ 。

现在要检验原假设 $H_0: F(x) = F_0(x)$ 。设在原假设 H_0 成立的条件下, 总体 X 落入区间 A_i 内的概率为 p_i , 即

$$p_i = P(A_i) = F_0(a_i) - F_0(a_{i-1}) \quad (i=1, 2, \dots, k)$$

那么, 此时 n 个观测值中, 恰有 n_1 个落入 A_1 中, n_2 个落入 A_2 中, \dots , n_k 个落入 A_k 中的概率为 $\frac{n!}{n_1! n_2! \dots n_k!} p_1^{n_1} p_2^{n_2} \dots p_k^{n_k}$, 这是一个多项分布。

按照大数定律, 在 H_0 为真时, 频率 $f(A_i) = \frac{n_i}{n}$ 与频率 p_i 的差异不应太大。根据这个思想,

皮尔逊构造了一个统计量, 即 $\chi^2 = \sum_{i=1}^k \frac{(n_i - np_i)^2}{np_i}$, 称作皮尔逊 χ^2 统计量。

当原假设 $H_0: F(x) = F_0(x)$ 为真时, 即 p_1, p_2, \dots, p_k 为总体的真实概率时, 皮尔逊 χ^2 统计量

$\chi^2 = \sum_{i=1}^k \frac{(n_i - np_i)^2}{np_i}$ 的渐近分布是自由度为 $k-1$ 的 χ^2 分布。

设 $F(x; \theta_1, \theta_2, \dots, \theta_m)$ 为总体的真实分布, 含有 m 个未知参数。在 $F(x; \theta_1, \theta_2, \dots, \theta_m)$ 中用 $\theta_1, \theta_2, \dots, \theta_m$ 的极大似然估计 $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_m$ 代替 $\theta_1, \theta_2, \dots, \theta_m$, 并且以 $F(x; \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_m)$ 来估计 p_i , 即

$\hat{p}_i = F(a_i; \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_m) - F(a_{i-1}; \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_m)$, 则统计量 $\chi^2 = \sum_{i=1}^k \frac{(n_i - n\hat{p}_i)^2}{n\hat{p}_i}$ 当 n 充分大时, 服从自

由度为 $k-m-1$ 的 χ^2 分布。

用总体分布假设的皮尔逊 χ^2 检验法的检验步骤如下:

(1) 把总体的值域划分成 k 个互不相容的区间 $A_i = (a_i, a_{i+1}]$ ($i=1, 2, \dots, k$)。

其中, a_1, a_{k+1} 可以分别取 $-\infty, +\infty$ (每个区间内必须包含不少于 5 个个体, 否则, 可把包含少于 5 个个体的区间并入其相邻的区间, 或把几个频数都少于 5 的但不一定相邻的区间并成一个区间)。

(2) 在 $H_0: F(x) = F_0(x)$ 为真下, 用极大似然估计总体分布所含的未知参数。

(3) 在 $H_0: F(x) = F_0(x)$ 为真下, 计算理论概率 $p_i = F_0(a_{i+1}) - F_0(a_i)$ ($i=1, 2, \dots, k$), 并计算出理论频数 np_i 。

(4) 按照样本观测值 x_1, x_2, \dots, x_n 落在区间 $A_i = (a_i, a_{i+1}]$ 中的个数, 即实际频数 n_i ($i=1, 2, \dots, k$)

和理论频数 np_i , 计算 $\frac{(n_i - np_i)^2}{np_i}$ 的值。

(5) 按照给定的显著性水平 α , 查自由度为 $k-m-1$ 的 χ^2 分布表得 $\chi_\alpha^2(k-m-1)$, 其中 m 是未知参数的个数。

(6) 若 $\chi^2 = \sum_{i=1}^k \frac{(n_i - np_i)^2}{np_i} \geq \chi_\alpha^2(k-m-1)$, 则拒绝 H_0 ; 否则, 接受 H_0 。

7.8 其他检验

7.8.1 秩和检验

如果两个总体 A 、 B 具有相同的概率分布, 分别从两个总体 A 、 B 中抽取大小为 n_1 和 n_2 的样本进行测定, 得到两组测定值:

$$\begin{aligned} x_1, x_2, \dots, x_{n_1} \\ y_1, y_2, \dots, y_{n_2} \end{aligned}$$

将两组测定值混合起来由小到大顺序排列, 每个测定值在序列中排列的次序, 称为该测定值的秩。一组样本测定值中各测定值的秩的总和, 称为改组测定值的秩和。当两组测定值中某个值相等时, 其秩等于相应两个测定值秩的平均值。如果两个总体具有相同的概率分布, 那么在混合排列的序列中第 i 个序次是测定值 x_i 或 y_i 的概率是相同的。

在 MATLAB 中, 使用 `ranksum` 函数进行两个分布一致性的检验——秩和检验, 其调用格式如下:

`p = ranksum(x,y)`: 返回两个分布一致性的检验, 其中 x 和 y 为两个独立的总体样本, 可以不等长; p 为两个总体样本 x 和 y 一致的显著性概率, 若 p 接近于 0, 则不一致较明显。

`[p,h] = ranksum(x,y)`: 返回的 h 为检验结果, $h=0$ 表示 x 和 y 的总体差别不显著; $h=1$ 表示 x 和 y 的总体差别显著。返回的 p 为产生两独立样本的总体是否相同的显著性概率 x 、 y 可为不等长向量, α 为默认的显著水平 0.05。

`[p,h] = ranksum(x,y,'alpha',alpha)`: 返回的 h 为检验结果, $h=0$ 表示 x 和 y 的总体差别不显示, $h=1$ 表示 x 和 y 的总体差别显著。返回的 p 为产生两独立样本的总体是否相同的显著性概率 x 、 y 可为不等长向量, α 为给定的显著水平。

`[p,h] = ranksum(...,'method',method)`: 根据 `method` 返回检验, `method` 可取 'exact' 和 'approximate', `exact` 表示精确运算, 一般在小样本中使用, 而 `approximate` 表示正态近似, 一般在大样本中使用。

`[p,h,stats] = ranksum(...)`: 返回的 h 为检验结果, $h=0$ 表示 x 和 y 的总体差别不显著; $h=1$ 表示 x 和 y 的总体差别显著。返回的 p 为产生两独立样本的总体是否相同的显著性概率 x 、 y 可为不等长向量, α 为给定的显著水平。stats 为统计构造的一些值, 包括如下:

(1) `ranksum`——检验统计样本的排序数值和。

(2) `zval`—— z 统计量的值 (一般只适用于大样本), 即取 $z = \frac{\bar{x} - u}{\sigma/\sqrt{n}}$, 服从标准的正态分布。

【例 7-27】某商店为了确定向公司 A 或公司 B 购买某种商品, 将 A 和 B 公司以往的各次进

货的次品率进行比较,如下所示,设两样本独立。问两公司的质量有无显著差异。设两公司商品的次品的密度最多只差一个平移,即 $\alpha=0.05$ 。

A: 7.0 3.6 9.2 6.7 5.3 10.4 5.0 3.7 9.9 8.7

B: 5.6 3.7 8.9 10.2 6.8 3.8 4.6 8.5 5.2 8.4 10.1 5.6 13.1

其实现的 MATLAB 程序代码如下:

```
>> clear all;
A=[7.0 3.6 9.2 6.7 5.3 10.4 5.0 3.7 9.9 8.7];
B=[5.6 3.7 8.9 10.2 6.8 3.8 4.6 8.5 5.2 8.4 10.1 5.6 13.1];
[p,h,stats]=ranksum(A,B,0.05)
```

运行程序,输出如下:

```
p =
    0.8523
h =
     0
stats =
    zval: -0.1861
    ranksum: 116.5000
```

结果表明:一方面,两样本总体均值相等的概率为 0.8523,不接近于 0;另一方面, $h=0$ 也说明可以接受原假设 H_0 ,即认为两个公司的商品质量无明显差异。

7.8.2 中值检验

在 MATLAB 的统计工具箱中提供了 signrank 函数和 signtest 函数实现中值检验。分别介绍如下。

1. signrank 函数

功能:对相等中位数的 Wilcoxon 符号检验。其调用格式如下:

```
p = signrank(x)
p = signrank(x,m)
p = signrank(x,y)
[p,h] = signrank(...)
[p,h] = signrank(...,'alpha',alpha)
[p,h] = signrank(...,'method',method)
[p,h,stats] = signrank(...)
```

说明:函数返回在显著水平下两对应样本 x 、 y 的中值相等的概率。 x 、 y 必须为两维数相同的向量; y 可为一数量,此情况下,函数计算 x 的中值与常数 y 不相等的概率; α 为给定的显著水平,它必须为 0 和 1 之间的数量。 $method$ 为返回的检验方法。 h 为返回假设检验的结果,如果 x 、 y 的中值差(区别于 0)不显著,则 h 为 0,否则为 1;如果原假设为真,则 p 为观察值等于或远大于原数据值的概率。如果 p 接近于 0,则可对原假设质疑; $stats$ 为统计构造的一些值。

【例 7-28】利用 signrank 函数进行中值检验。

其实现的 MATLAB 程序代码如下:

```
>> clear all;
before = lognrnd(2,.25,10,1);
```

```
after = before+trnd(2,10,1);
[p,h,stats] = signrank(before,after)
```

运行程序，输出如下：

```
p =
    0.3223
h =
     0
stats =
    signedrank: 17
```

2. signtest 函数

功能：用于符号检验。其调用格式如下：

```
p = signtest(x)
p = signtest(x,m)
p = signtest(x,y)
[p,h] = signtest(...)
[p,h] = signtest(...,'alpha',alpha)
[p,h] = signtest(...,'method',method)
[p,h,stats] = signtest(...)
```

说明：函数返回在显著水平下两对应样本 x 、 y 的中值相等的概率。 x 、 y 必须为两维数相同的向量； y 可为一数量，此情况下，函数计算 x 的中值与常数 y 不相等的概率； α 为给定的显著水平，它必须为 0 和 1 之间的数量。 $method$ 为返回的检验方法。 h 为返回假设检验的结果，如果 x 、 y 的中值差不显著地区别于 0，则 h 为 0；如果 x 和 y 的中值差显著地区别于 0，则为 1；如果原假设为真，则 p 为观察值等于或远大于原数据值的概率。如果 p 接近于 0，则可对原假设质疑； $stats$ 为统计构造的一些值。

【例 7-29】利用 signtest 函数进行中值检验。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
before = lognrnd(2,.25,10,1);
after = before + (lognrnd(0,.5,10,1) - 1);
[p,h,stats] = signtest(before,after)
```

运行程序，输出如下：

```
p =
    0.7539
h =
     0
stats =
    sign: 4
```


第8章 回归分析

8.1 概述

在许多问题中，常常会遇到许多相互联系、相互制约的变量，常见的变量之间的关系有两类：一类是确定性的关系（或称函数关系），例如物体作匀速运动时，速度 v 、时间 t 及路程 s 之间有 $s=vt$ 的确定性关系；又如一段电路中，电阻为 R 与电路两端的电压 U 及电流 I 之间由欧姆定律 $U=IR$ 确定；等等。另一类为非确定性关系，它们之间虽有一定的关系却又不完全确定。如人的血压与年龄、身高与体重的关系。一般来说，人的年龄越大血压就越高；身材越高，体重越重。但是年龄相同者，血压未必相同；身高相同者，体重也未必相同。又如同样收入的家庭，用于食品的消费支出往往不相同；等等。这些变量之间的共同特点是，虽然他们有一定的关系，但又不能用确定的函数关系来表达，这样的关系叫做相关关系。回归分析就是研究这种相关关系的一种统计方法。

在相关关系中，有些变量，例如上面提到的人的年龄、身高、家庭的收入等，都是可以在某一范围内确定数值的，这些变量称为可控变量或自变量；而可控变量取定后，与它们对应的人的体重、血压、消费水平的取值虽然可观察但不可控制，这类变量称为随机变量或因变量。

“回归”一词是由美国的高尔顿于 1886 年首先提出的，他在研究家族成员之间的遗传规律时发现：虽然高个子的父亲确有生高个子儿子的趋向，但一群高个子父亲的儿子的平均身高却低于父亲们的身高；反之，一群矮个子父亲的儿子们的平均身高却高于父亲们的平均身高。高尔顿称这一现象为“向平均高度的回归”，也即回归到“平均祖先型”。今天人们对“回归”这一概念的理解与高尔顿的原意已有很大不同，但这一名词一直沿用下来，成为统计学中最常用的概念之一。

研究一个随机变量与一个（或几个）可控变量之间的相关关系的统计方法称为回归分析。只有一个自变量的回归分析叫做一元回归分析，多于一个自变量的回归分析叫做多元回归分析。

回归分析是研究一个随机变量与一个（或多个）普通变量之间相互关系的统计方法。它主要解决以下几个方面的问题：

(1) 从一组数据出发，确立变量间是否存在相关关系，如果存在相关关系，确定它们之间合适的数学表达式即经验公式或回归方程，并对它的可信程度作统计检验。

(2) 从共同影响一个变量的许多变量中，判断哪些变量的影响是显著的，哪些变量的影响是不显著的。

(3) 利用所确定的回归方程进行预测和控制。

回归分析的应用很广泛，在工农业生产和科学实验中许多问题都可以用这种方法得到解决。例如在各种预报预测中，习惯上称预报对象这个随机变量为因变量，称与此有关的非随机变量为自变量（或预报因子），应用回归分析可建立因变量（预报对象）与一组自变量（或一组预报因子）之间的数学表达式。又如在寻求生产过程的工艺最优化问题中，先要寻求工艺的

优化区域，在还不完全了解生产过程的物理、化学、生物等原理及经验的条件下，用回归分析来解决生产过程的最优化问题也是一个比较有效的数学方法。对于诸如经验公式的求得、因子分析、产品质量控制等，都要用到回归分析这一工具。

8.2 一元线性回归分析

8.2.1 一元线性回归分析数学模型

在一元线性回归分析中，通常考虑两个变量：一个是自变量 x ，其值是可以控制或精确测量的，认为它是非随机变量；另一个是因变量 y ，对给定的 x 值， y 的取值事先不能确定，故 y 是随机变量。为了研究 y 与 x 之间的相关关系，首先就要对变量 (x,y) 进行观测，收集数据。为使下面的讨论直观，先来考察一个例子。

【例 8-1】我们知道，营业税税收总额 y 与社会商品零售总额 x 有关。为能从社会商品零售总额去预测税收总额，需要了解两者的关系。现收集了如下 9 组数据，见表 8-1。

表 8-1 社会商品零售总额与税收总额

序 号	社会商品零售总额 x	营业税税收总额 y
1	142.08	3.93
2	177.30	5.96
3	204.68	7.85
4	242.88	9.82
5	316.24	12.50
6	341.99	15.55
7	332.69	15.79
8	389.29	16.39
9	453.40	18.45

试分析税收总额 y 与商品零售总额 x 的相关关系，建立回归方程。

通常将上述数据记为 $(x_i, y_i) (i=1, 2, \dots, n)$ ，本例 $n=9$ 。为了直观起见，可将这 n 对数据作为平面直角坐标系 xOy 中的 n 个点，通过描点在平面上得到一张“散点图”，以观察两个变量之间的线性相关性。本例的散点图如图 8-1 所示。

观察 n 个点在图中的散布情况，发现本例的 9 个点散布在一条直线附近。

可以这样理解图中的信息：税收总额 y 与商品零售总额 x 之间似乎存在一种线性关系，也就是说，税收总额 y 应当是商品零售总额 x 的线性函数；但是实际观察到的数据点 $(x_1, y_1), \dots, (x_9, y_9)$ 却不在一条直线上，这应当是未知的随机因素干扰的结果。换句话说，税收总额的观测值 y 由两部分叠加而成：一是税收总额 y 随商品零售总额 x 的变化而呈线性变化的趋势（用 $\alpha + \beta x$ 表示）；另一是其他随机因素干扰的总和（用 ε 表示），即观测数据 $(x_i, y_i) (i=1, 2, \dots, n)$ 应当满足关系式

$$y_i = \alpha + \beta x_i + \varepsilon_i \quad (8-1)$$

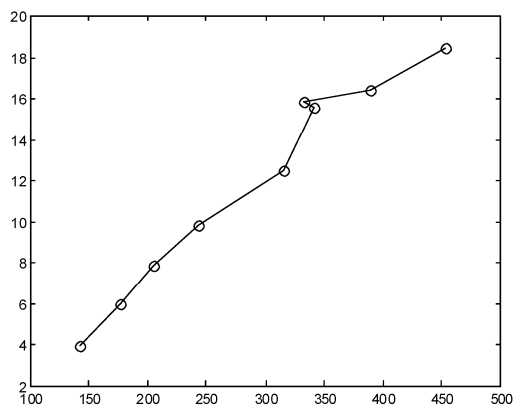


图 8-1 社会商品零售总额与税收总额(x,y)散点图

通常假定 $\varepsilon_i \sim N(0, \sigma^2) (i=1, 2, \dots, 9)$ 且各个 ε_i 相互独立。至此, 可以给出一元线性回归分析的基本概念。

设 x 是自变量 (非随机变量, 其值是可以控制或精确测量的), y 是因变量 (随机变量, 对给定的 x 值不能事先确定 y 的取值), 则称

$$y = \alpha + \beta x + \varepsilon \quad (\varepsilon \sim N(0, \sigma^2)) \quad (8-2)$$

为一元线性回归模型 (理论模型)。其中: α 、 β 称为模型参数; ε 称为模型随机误差。

线性函数

$$E(y) = \alpha + \beta x \quad (8-3)$$

的经验回归方程

$$\hat{y} = \hat{\alpha} + \hat{\beta} x \quad (8-4)$$

称为建立一元线性回归模型。其中: \hat{y} 是 $E(y)$ 的统计估计; $\hat{\alpha}$ 、 $\hat{\beta}$ 分别是 α 、 β 的统计估计, 称为经验回归系数。

设数据对 $(x_i, y_i) (i=1, 2, \dots, n)$ 是变量对 (x, y) 的观测数据, 则

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

称为一元样本回归方程 (数据模型)。其中, $\varepsilon_i \sim N(0, \sigma^2) (i=1, 2, \dots, n)$ 且各个 ε_i 相互独立。

一元线性回归就是寻求两个变量间的线性统计回归分析, 若其相关关系的统计规律性呈线性关系, 则称一元线性回归分析。其回归模型为

$$\hat{y} = a + bx \quad (8-5)$$

式中: y 为因变量; x 为自变量; a 和 b 为待定参数 (a 为常数项, b 为回归系数)。

8.2.2 参数的最小二乘估计

求回归方程(8-5)即要求出 α 与 β 的估计值 a 和 b , 使其对一切 x_i 观测值 y_i 与回归值 $\hat{y} = a + bx_i$ 的偏离达到最小。为此, 采用最小二乘法来求 α 与 β 的估计。令

$$e_i = y_i - (\alpha + \beta x_i) \quad (8-6)$$

通常采用最小二乘法来求参数 a 和 b 的估计, 即使得

$$Q(\alpha, \beta) = \sum e_i^2 \quad (8-7)$$

达到最小。

α 、 β 的最小二乘估计是指使下式成立的 a 和 b ：

$$Q(a, b) = \min_{\alpha, \beta} Q(\alpha, \beta) \quad (8-8)$$

即

$$Q(a, b) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - a - bx_i)^2 \quad (8-9)$$

由 a 和 b 所确定的直线 $\hat{y} = a + bx_i (i=1, 2, \dots, n)$ 称为 y 对 x 的线性回归方程。它表述了变量 y 同给定的变量 x 诸值间的平均变动关系。 y 对 x 的回归系数 b 表示 x 每变动一个单位所引起的 y 的平均变动。

由于 $\sum e_i^2$ 是 α 、 β 的二次函数且非负，所以存在最小值，根据微积分学的极值定理，可求解得到

$$\begin{cases} \left. \frac{\partial Q}{\partial a} \right|_{a=a, \beta=b} = -2 \sum_{i=1}^n (y_i - a - bx_i) = 0 \\ \left. \frac{\partial Q}{\partial \beta} \right|_{a=a, \beta=b} = -2 \sum_{i=1}^n (y_i - a - bx_i)x_i = 0 \end{cases} \quad (8-10)$$

把式 (8-10) 化简得

$$\begin{cases} na + b \sum_{i=1}^n x_i = \sum_{i=1}^n y_i \\ a \sum_{i=1}^n x_i + b \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i \end{cases} \quad (8-11)$$

把式 (8-11) 称为正规方程组。由此得

$$\begin{cases} \hat{a} = \bar{y} - b\bar{x} \\ \hat{b} = \frac{\sum_{i=1}^n x_i y_i - \frac{1}{n} \left(\sum_{i=1}^n x_i \right) \left(\sum_{i=1}^n y_i \right)}{\sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2} \end{cases} \quad (8-12)$$

为简化记号，令

$$l_{xy} = \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^n x_i y_i - \frac{1}{n} \sum_{i=1}^n x_i \cdot \sum_{i=1}^n y_i \quad (8-13)$$

$$l_{xx} = \sum_{i=1}^n (x_i - \bar{x})^2 = \sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2 \quad (8-14)$$

其中 $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ ， $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ 。故最小二乘估计为

$$\begin{cases} a = \bar{y} - b\bar{x} \\ b = \frac{l_{xy}}{l_{xx}} \end{cases} \quad (8-15)$$

在平面直角坐标系上, 通过 $(0, a)$ 与 (\bar{x}, \bar{y}) 两点引一直线即为所求的回归直线。这是因为点 $(0, a)$ 显然在直线

$$\hat{y} = a + bx$$

上。又若将 $a = \bar{y} - b\bar{x}$ 代入上式, 则有

$$\hat{y} - \bar{y} = b(x - \bar{x}) \quad (8-16)$$

8.2.3 回归显著性检验

给定以上模型和实测数据以后, 总可以得到待定参数的拟合值, 但由此确定的回归方程式不一定有意义。因此, 需要对得到的回归系数做显著性检验, 即检验回归系数是否为 0, 如果为 0, 则说明因变量与自变量无关, 回归方程无意义。回归系数的显著性检验有多种方法, 下面介绍 F 检验和相关系数检验法。

记

$$\begin{aligned} SST &= \sum_{i=1}^n (y_i - \bar{y})^2 \\ SSR &= \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 \\ SSE &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \end{aligned}$$

则有

$$SST = SSR + SSE$$

式中: **STT** 为观测值 y_i 与均值 \bar{y} 的总离差平方和; **SSR** 为回归值 \hat{y}_i 与均值 \bar{y} 的总离差平方和, 这是由回归所能解释的部分; **SSE** 为观测值 y_i 与回归值 \hat{y}_i 的总离差亦即残差平方和, 这是不能由回归加以解释的部分。

若回归方程有意义, 即 y 波动主要是由 x 变化引起的, 其他一切因素是次要的。则要求 **SSR** 尽可能大, 而 **SSE** 尽可能小。

1. r 检验法

变量 y 的各个观测值点聚在回归直线 $\hat{y} = a + bx$ 周围的紧密程度, 称作回归直线对样本数据点的拟合程度, 通常用可决系数 (亦称测定系数) r^2 来表示。

显然, 变量 y 的各个观测值点与回归直线越靠近, S_R 在 S_T 中所占的比重比大, 因而定义

$$r^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (8-17)$$

它可以用来测定回归直线对各观测点的拟合程度。若全部观测值点 $y_i (i=1, 2, \dots, n)$ 都落在回归直线上, 则剩余平方和 $SSE=0$, $r^2=1$; 若 x 完全无助于解释 y 的偏差, 则回归平方和 $SSR=0$, $r^2=0$ 。显然, r^2 越接近于 1, 用 x 的变化解释 y 的偏差的部分就越多, 表明回归直线与各观测值

点越接近, 回归线的拟合程度越高。可决系数 r^2 在 $[0, 1]$ 上取值。

回归直线对样本数据点拟合程度的另一测度是线性相关系数 r 。在一元线性回归中, 线性相关系数 r 实际上是可决系数 r^2 的平方根, 即 $r = \pm\sqrt{r^2}$, r 的正负号与回归系数 b 的正负号相同, $|r|$ 越接近于 1, 表明回归直线对样本数据点的拟合程度越高。

2. F 检验法

在 $H_0: b=0$ 为真时, 可证明

$$F = \frac{SSR}{SSE/(n-2)} \sim F(1, n-2)$$

当 H_0 为真时, $\frac{SSR}{SSE/(n-2)}$ 有变大趋势, 因而 F 也有变大趋势, 故应取单侧拒绝域。对给

定的显著性水平 α , 当 $F \geq F_{\alpha}(1, n-2)$ 时, 认为 $b=0$ 不真, 称方程是显著的。反之, 方程不显著。

这种用 F 检验对回归方程作显著性检验的方法称为方差分析。其检验过程可由一张“方差分析表”来进行, 见表 8-2。

表 8-2 回归方程显著性检验的方差分析

方差来源	偏差平方和	自由度	F 值	p 值	显著性
回归	SSR	$f_R=1$	$F = \frac{\frac{SSR}{f_R}}{\frac{SSE}{f_E}}$	$p=P\{F(1, n-2)>F\}$	
残差	SSE	$f_E=n-2$			
总计	SST	$f_T=n-1$			

通常, 若 $F \geq F_{\alpha}(1, n-2)$, 则为高度显著; 若 $F_{0.05}(1, n-2) \leq F < F_{0.01}(1, n-2)$, 则为显著; 若 $F < F_{0.05}(1, n-2)$, 则为不显著。

3. 估计标准误差

可决系数 r^2 和线性相关系数 r 描述回归直线对样本数据点的拟合程度, 但没有表示出变量 y 的诸观测值 y_i 与回归直线 $\hat{y}_i = a + bx_i$ 的绝对离差数额。定义

$$S_y^{2\text{def}} \text{MSE} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n-2} = \frac{\sum_{i=1}^n e_i^2}{n-2} \quad (8-18)$$

为最小二乘残差值 e_i 方差, 则称

$$S_y = \sqrt{\text{MSE}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n-2}} = \sqrt{\frac{\sum_{i=1}^n e_i^2}{n-2}} \quad (8-19)$$

为变量 y 对 x 的最小二乘回归的估计标准误差, 简称估计标准误差。 S_y^2 和 S_y 可以作为诸 y 值与回归直线变差的测度。 S_y 的计量单位与变量 y 的单位相同。显然, S_y 越小表明误差越小。

8.2.4 回归方程的预测

利用变量 x 与 y 的 n 对样本数据建立的回归方程

$$\hat{y} = a + bx$$

如果通过了上述的各种检验, 即可用来预测。预测问题就是在确定自变量的某一个 x_0 值时求相

应的因变量 y 的估计值, 其中又可以分为点预测和区间预测。

1. 点预测

将自变量的预测值 x_0 代入回归方程式 $\hat{y} = a + bx$ 所得到的因变量 y 的值 \hat{y}_0 , 作为与 x_0 相对应的 y_0 的预测值, 就是点预测。可以证明 \hat{y}_0 是无偏预测。

2. 区间预测

对于与 x_0 相对应的 y_0 , $\hat{y}_0 = a + bx_0$ 可以作为 $y_0 = \alpha + \beta x_0 + \varepsilon$ 的一个点估计值。但不同的样本会得到不同的 α 、 β , 因此, \hat{y}_0 与 y_0 之间总存在一定的抽样误差。在回归模型的假设条件下, 可以证明

$$(\hat{y}_0 - y_0) \sim N \left[0, \sigma^2 \left[1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_i (x_i - \bar{x})^2} \right] \right]$$

因此, y_0 的概率为 $1-\alpha$ 的预测区间为

$$\hat{y}_0 \pm t_{\frac{\alpha}{2}} \sigma \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_i (x_i - \bar{x})^2}}$$

当 x_0 取值在 \bar{x} 附近, n 比较大时, 可以近似地认为 $(\hat{y}_0 - y_0) \sim N(0, S_y^2)$ 。因而 y_0 的概率为 $1-\alpha$ 的预测区间为

$$\hat{y}_0 \pm t_{\frac{\alpha}{2}} S_y$$

注意: 实际应用时, 常常采用这一区间作为因变量 y 对应于自变量 x_0 的回归预测区间。

当 $\alpha=0.05$ 时, y_0 的 95% 的预测区间为 $\hat{y}_0 \pm 2S_y$; 当 $\alpha=0.01$ 时, y_0 的 99% 的预测区间为 $\hat{y}_0 \pm 3S_y$ 。

8.2.5 一元线性回归函数介绍

关于线性回归分析的 `rcoplot` 函数在第 5 章已经介绍过, 在此不再介绍此两个函数, 只分别通过一个例子来温故。对其他的线性回归函数进行介绍。

【例 8-2】测定了苯甲腈和 9 个取代苯甲腈化合物的有机碳吸附系数 K_{oc} 和正辛醇—水分配系数 K_{ow} , 试找出二者之间的关系。数据见表 8-3。

表 8-3 $\lg K_{oc}$ 与 $\lg K_{ow}$ 数据

化合物编号	$\lg K_{oc}$	$\lg K_{ow}$	化合物编号	$\lg K_{oc}$	$\lg K_{ow}$
1	2.15	1.60	6	2.44	2.25
2	2.22	1.59	7	2.52	2.24
3	2.30	1.45	8	2.52	2.24
4	2.29	1.57	9	2.58	2.27
5	2.31	1.84	10	2.60	2.45

其实现的 MATLAB 程序代码如下:

```
>> clear all;
x=[1.60 1.59 1.45 1.57 1.84 2.25 2.24 2.24 2.27 2.45]';
y=[2.15 2.22 2.30 2.29 2.31 2.44 2.52 2.52 2.58 2.60]';
```

`rstool(x,y)` %`regress` 函数不包括常数项,所以采用此函数

运行程序,效果如图 8-2 所示的线性拟合图。在列表框中选择 `linear`, 在命令窗口输入下列命令, 则得到拟合函数的系数、均方差和残差。

```
>> beta'      %拟合系数
ans =
    1.6443     0.3845
>> rmse       %均方差
ans =
    0.0695
>> residuals' %残差
ans =
   -0.1096   -1.0367    0.0984    0.620   -0.0503   -0.695    0.143    0.0143
0.0628    0.0136
>> d=x2fx(x)  %转换成设计矩阵
d =
    1.0000    1.6000
    1.0000    1.5900
    1.0000    1.4500
    1.0000    1.5700
    1.0000    1.8400
    1.0000    2.2500
    1.0000    2.2400
    1.0000    2.2400
    1.0000    2.2700
    1.0000    2.4500
>> regress(y,d)
ans =
    1.6352
    0.3886
```

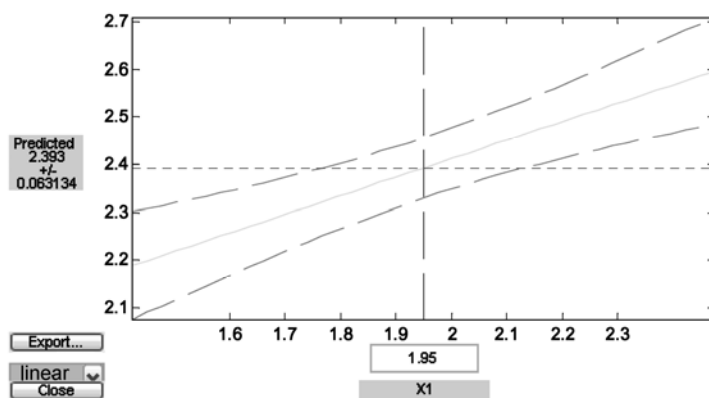


图 8-2 线性拟合

1. `regress` 函数

功能: 进行线性回归分析。其调用格式如下:

`b = regress(y,X)`: 根据输入参数 `y` 与 `X`, 用最小二乘法求线性回归系数 `b`。

`[b,bint,r,rint,stats]=regress(y,X)`: 返回参数 b 的 95% 置信区间 `bint`; 返回残差 r 以及残差 95% 置信区间 `rint`; 返回值 `stats` 是一个有三个分量的向量, 其中包含可决系数 r^2 、 F 值以及回归的 p 值。

`[...]=regress(y,X,alpha)`: 设置输入参数 `alpha`。可以求置信度是 $100(1-\alpha)\%$ 的置信区间。

【例 8-3】某种合金强度与碳含量有关, 研究人员在生产试验中收集了该合金的强度 y 与碳含量 x 的数据 (表 8-4)。试建立 y 与 x 的函数关系模型。并检验模型的可信度, 检查数据中是否有异常点。

表 8-4 合金的强度与碳含量数据表

x	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.20	0.21	0.23
y	42.0	41.5	45.0	45.5	45.0	47.2	49.0	55.0	50.3	55.0	55.5	60.5

其实现的 MATLAB 程序代码如下:

```
>> clear all;
% 数据的输入
x1=0.1:0.01:0.18;
x2=[x1 0.2 0.21 0.23]';
y=[42.0 41.5 45.0 45.5 45.0 47.2 49.0 55.0 50.3 55.0 55.5 60.5]';
x=[ones(12,1),x2];
% 作数据的散点图
plot(x2,y,'rp');
% 回归分析
[b,bint,r,rint,stats]=regress(y,x);
b,bint,stats
% 作残差分析图
rcoplot(r,rint);
title('残差图的绘制');
xlabel('数据'); ylabel('残差');
% 预测及作回归线图
z=b(1)+b(2)*x2;
figure;
plot(x2,y,'rp',x2,z,'b');
```

运行程序, 输出如下:

```
b =
    26.9502
   141.1041
bint =
    22.3110    31.5894
   112.6683   169.5399
stats =
    0.9244   122.2457    0.0000    3.0240
```

残差图如图 8-3 所示, 散点图及回归线图如图 8-4 所示。

2. leverage 函数

功能: 生成回归的中心化杠杆值, 以衡量由于给定观察值在输入空间中的位置而引起的对回归的影响。其调用格式如下:

`h = leverage(data)`: 找到 `data` 矩阵中每一行的中心化杠杆值。

$h = \text{leverage}(\text{data}, \text{model})$: 找到回归的中心化杠杆值, 并使用 'model' 指定的模型类型。'model' 可以是以下的任何一种。

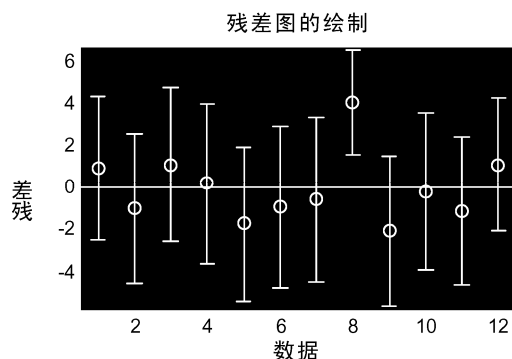


图 8-3 残差图

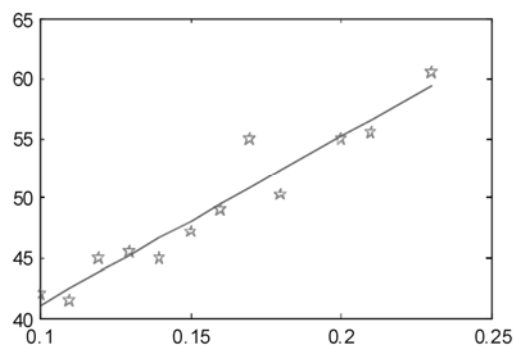


图 8-4 散点图及回归直线

- (1) interaction: 包含常数项、线性项和交互项。
- (2) quadratic: 交互项加上平方项。
- (3) purequadratic: 包含常数项、线性项和平方项。

【例 8-4】leverage 函数用法示例。

```
>> clear all;
>> load hald %MATLAB 自带的数据库
h = max(leverage(ingredients, 'linear'))
```

运行程序, 输出如下:

```
h =
    0.7004
```

3. regstats 函数

功能: 回归诊断图形用户界面。其调用格式如下:

```
regstats(y,X,model)
```

```
stats = regstats(...)
```

```
stats = regstats(y,X,model,whichstats)
```

说明: 对带常数项的线性回归模型进行回归诊断, 可以创建一个图形窗口并控制回归模型的级次。

【例 8-5】regstats 函数示例。

```
>> whichstats = {'yhat','r'};
stats = regstats(heat,ingredients,'linear',whichstats)
yhat = stats.yhat;
r = stats.r
```

运行程序，输出如下：

```
stats =
    source: 'regstats'
      yhat: [13x1 double]
       r: [13x1 double]

r =
    0.0047604
    1.5112
   -1.6709
   -1.7271
    0.25076
    3.9254
   -1.4487
   -3.175
    1.3783
    0.28155
    1.991
    0.97299
   -2.2943
```

4. stepwise 函数

功能：逐步回归的交互式环境。其调用格式如下：

stepwise：返回逐步回归交互式环境图形。

stepwise(X,y)：拟合 y 与 X 的列之间的回归模型。

stepwise(X,y,inmodel)：inmodel 是 X 中的选定系数，包含在初始模型中。

stepwise(X,y,inmodel,penter,premove)：允许控制拟合系数的置信区间宽度。

【例 8-6】stepwise 函数示例。

```
>> stepwise
```

运行程序，效果如图 8-5 所示。

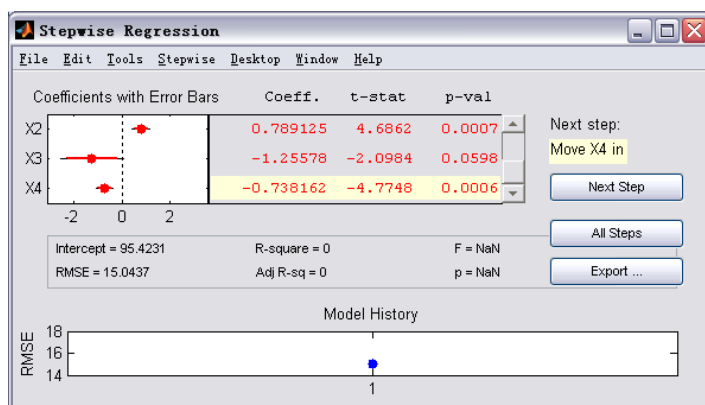


图 8-5 返回当前交互式环境界面

5. glmfit 函数

功能：广义线性拟合。其调用格式如下：

`b = glmfit(X,y,distr)`: 拟合 y 与 X 的列之间的回归模型。

`b = glmfit(X,y,distr,param1,val1,param2,val2,...)`: 对于拟合提供其他控制。

`[b,dev,stats] = glmfit(...)`: 返回附加输出 `dev` 和 `stats`。

【例 8-7】广义线性拟合示例。

```
>> clear all;
x = [2100 2300 2500 2700 2900 3100 ...
     3300 3500 3700 3900 4100 4300]';
n = [48 42 31 34 31 21 23 23 21 16 17 21]';
y = [1 2 0 3 8 8 14 17 19 15 17 21]';
b = glmfit(x,[y n],'binomial','link','probit');
yfit = glmval(b, x,'probit','size', n);
plot(x, y./n,'rp',x,yfit./n,'-', 'LineWidth',2)
```

运行程序，效果如图 8-6 所示。

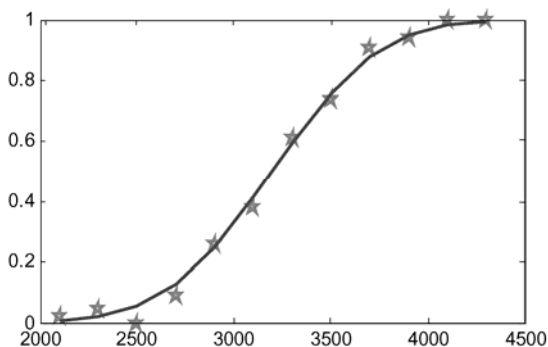


图 8-6 数据广义线性拟合效果

6. robust 函数

功能：稳健回归。其调用格式如下：

`b=robust(x, y)`: 用稳健回归来拟合，它受异常值的影响小。

`[b, stats]= robust(x, y)`: 返回 `stats` 结构。

`[b, stats]= robust(x, y, 'wfun', 'tune', 'cost')`: 指定一个加权函数、一个协调常数和是否显示常数项。

7. rstool 函数

功能：交互式拟合及响应面的可视化。其调用格式如下：

`rstool(X,Y,model)`: 显示 95% 置信区间的交互式预测图，并允许控制初始回归模型。

`rstool(x,y,model,alpha,xname,yname)`: 在图中标注 x 轴和 y 轴。

【例 8-8】交互式拟合及响应面的可视化示例。

```
>> load reaction          %MATLAB 自带的数据库
alpha = 0.01; % Significance level
rstool(reactants,rate,'quadratic',alpha,xn,yn)
```

运行程序，效果如图 8-7 所示。

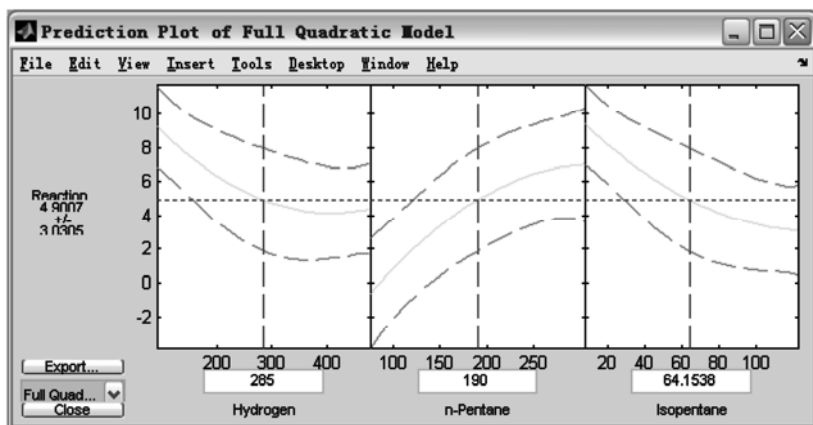


图 8-7 响应面的可视化效果

8. x2fx 函数

功能：将一个因子设置矩阵转换为一个设计矩阵。其调用格式如下：

$D = \text{x2fx}(X, \text{model})$ ：为带常数项线性模型，将系统输入矩阵转换为设计矩阵，并允许控制回归模型的级次。

【例 8-9】x2fx 函数应用示例。

```
>> X = [1 10
        2 20
        3 10
        4 20
        5 15
        6 15];
model = [0 0
        1 0
        0 1
        1 1
        2 0];
D = x2fx(X, model)
D =
     1     1    10    10     1
     1     2    20    40     4
     1     3    10    30     9
     1     4    20    80    16
     1     5    15    75    25
     1     6    15    90    36
```

8.2.6 一元线性回归分析的编程实现

进行一元线性回归分析，通常是先考察所给数据的散点图中的数据点是否大致在一条直线上。如果数据点大致在一条直线上，则求出其回归线的方程，并以此进行预测。

1. 检验函数

要进行一元线性回归分析，需要检验所输入的横坐标与纵坐标的数据点的个数是否相等，以防止运行程序时出错。为此，可以用下面的检验函数进行检验。

其源代码如下：

```
function [X,Y]=jyhshu(x,y)
X=[];
Y=[];
[m,n]=size(x);
if min(m,n)~=1
    error('第一个输入参数 x 必须是向量');
end
[m,n]=size(y);
if min(m,n)~=1
    error('第二个输入参数 y 也必须是向量');
end
X=x(:);
Y=y(:);
if length(X)~=length(Y)
    error('两个输入向量的长度必须相等');
end
```

2. 散点图

根据所给的数据，绘制散点图，只要用二维绘图函数 `plot` 即可。考虑到输入数据时，有可能出现错误，可以用下面的散点图函数，先调用检验函数，检验输入参数确实为两个长度相等的向量，再绘制散点图。

其源代码如下：

```
function sdtu(x,y)
[x,y]=jyhshu(x,y); %检验两个向量是否相等
figure
plot(x,y,'r');
title('绘制散点图');
```

这一函数创建图形窗口，绘制散点图后，再加上标题“绘制散点图”。

3. 回归系数

如果所给数据的散点图上的数据点大致在一条直线上，也就是说，变量的线性回归明显，可以用下面的函数求出回归方程的系数：

```
function [a,b]=hgxshu(x,y)
[x,y]=jyhshu(x,y) %检验两个向量是否相等
n=length(x);
a=[ones(n,1),x];
ab=a\y;
a=ab(1);
b=ab(2);
```

通常一元线性回归方程是 $y = a + bx$ 形式，所以计算系数时，要补充一个全 1 的列向量。函数的输出参数就是回归方程的系数。

4. 回归线

用下面的函数可以在一个图形窗口中绘制一元线性回归的散点图和回归线图：

```
function hgxian(x,y)
```

```
[a,b]=hgxshu(x,y); % 计算回归系数
```

```
X=[min(x),max(x)];
```

```
Y=a+b.*X;
```

```
figure
```

```
plot(x,y,'rp',X,Y,'b');
```

```
title('绘制回归线图');
```

通过回归线和散点的拟合情况，可以直观地看出拟合的效果。

5. 一元线性回归的检验

对一元线性回归可以用方差分析、可决系数、标准误差来检验拟合程度的好坏。其调用函数如下：

```
function [table,r2,Sy]=yyxhjdjyan(x,y,alpha)
```

```
if nargin<3
```

```
    alpha=[0.05,0.01];
```

```
end
```

```
[a,b]=hgxshu(x,y); % 计算回归系数
```

```
yhat=a+b*x;
```

```
SSR=(yhat-mean(y))*(yhat-mean(y));
```

```
SSE=(yhat-y)*(yhat-y);
```

```
SST=(y-mean(y))*(y-mean(y));
```

```
n=length(x);
```

```
Fb=SSR/SSE*(n-2)
```

```
Falpha=finv(1-alpha,1,n-2);
```

```
table=cell(4,7);
```

```
table(1,:)={'方差来源','偏差平方和','自由度','方差','F 值','Fa 值','显著性'};
```

```
table(2,1:6)={'回归',SSR,1,SSR,Fb,min(Falpha)};
```

```
table(3,1:6)={'剩余',SSE,n-2,SSE/(n-2),[],max(Falpha)};
```

```
table(4,1:3)={'总和',SST,n-1};
```

```
if Fb>=max(Falpha)
```

```
    table{2,7}='高度显著';
```

```
elseif (Fb<max(Falpha))&(Fb>=min(Falpha))
```

```
    table{2,7}='显著';
```

```
else
```

```
    table{2,7}='不显著';
```

```
end
```

```
r2=SSR/SST;
```

```
Sy=sqrt(SSE/(n-2));
```

这一函数的输出参数是：方差分析表 table、可决系数 r2 和估计的标准误差 Sy。

对回归方程的显著水平，默认值是 0.05 和 0.01。显然水平的值可以根据具体的问题用第三个输入参数 alpha 来设定。

6. 预测

线性回归一个重要应用是进行预测，用下面的程序可以根据输入数据，进行点预测和区间预测：

```
function [varargout]=yuce(x,y,xx,pp)
```

```
if(nargin<4)|(~isnumeric(pp))|(pp<=0)|(pp>=1)
```

```

    pp=0;
end
[X,Y]=jyhshu(x,y);           % 检验两个向量是否相等
[a,b]=hgxsu(x,y);           % 调用函数,计算回归系数
[table,r2,Sy]=yyxhjdjyan(x,y); % 调用函数
n=length(x);
yy=a+b.*xx;                 % 点预测
ta=tinv(0.5+pp/2,n);
yl=yy-ta.*Sy;
yr=yy+ta.*Sy;
ylr=[yl(:),yr(:)];         % 区间预测
if pp<=0
    ylr=[];
end
if nargin==0
    disp('预测值是:');
    disp(num2str(yy(:)))
    if nargin==4
        disp('预测区间是:');
        disp(ylr)
    end
end
if nargin>0
    varargout{1}=yy(:);
end
if nargin>=2
    varargout{2}=ylr;
end

```

这一函数能够根据输入的原始数据 x 、 y ，以及要进行预测的数据 xx 进行点预测。如果给出了预测的概率（第 4 个输入参数 pp ），还可以进行区间预测。如果没有给出预测的概率，则只进行点预测。

7. 一元线性回归总体分析

利用下面的函数可以对一元回归问题进行总体分析：

```

function [a,b,table,r2,Sy,yy,ylr]=regress1(x,y,xx,pp)
if nargin<3
    yy=[];
    ylr=[];
end
if(nargin<4)|(~isnumeric(pp))|(pp<=0)|(pp>=1)
    pp=0;
end
[x,y]=jyhshu(x,y);           % 检验两个向量是否相等
[a,b]=hgxsu(x,y);           % 调用函数,计算回归系数
[table,r2,Sy]=yyxhjdjyan(x,y); % 调用函数,计算误差

```



```

if nargin>=3
    [yy,ylr]=yuce(x,y,xx,pp);    %进行预测
end
hgxian(x,y)

```

这一函数调用了前面的各个函数，计算各种数据，并绘制回归线图。

【例 8-10】某市电子工业公司有 14 个所属企业，各企业年设备能力与年劳动生产率统计数据见表 8-5。试分析企业年设备能力与年劳动生产率的关系。估计该公司的设备能力从 7.1kW/人~7.6kW/人，其劳动生产率的可能值是多少？若该公司计划新建一个设备能力为 9.2kW/人的企业，估计劳动生产率及其 95%置信区间将为多少？

表 8-5 年设备能力与劳动生产率的关系表

企业	年设备能力 /(kW/人)	年劳动生产率 /(千元/人)	企业	年设备能力 /(kW/人)	年劳动生产率 /(千元/人)
1	2.8	6.8	8	4.9	9.8
2	2.9	6.8	9	5.4	10.6
3	3.0	6.7	10	5.3	10.7
4	2.7	7.3	11	5.4	11.1
5	3.5	7.0	12	6.8	11.8
6	3.6	8.9	13	7.2	12.1
7	4.1	9.2	14	7.0	12.4

可以用函数 regress1 一次性地把所有问题分析清楚。其实现的 MATLAB 程序代码如下：

```

>> clear all;
x=[6.8 6.8 6.7 7.3 7.0 8.9 9.2 9.8 10.6 10.7 11.1 11.8 12.1 12.4];
y=[2.8 2.9 3.0 2.7 3.5 3.6 4.1 4.9 5.4 5.3 5.4 6.8 7.2 7.0];
xx1=(7.1:0.1:7.6);
[a,b,table,r2,Sy,yy1]=regress1(x,y,xx1)

```

运行程序，输出如下：

```

a =
    -2.2393
b =
     0.7313
table =
Columns 1 through 5
    '方差来源'    '偏差平方和'    '自由度'    '方差'    'F 值'
    '回归'        [    31.8170]    [     1]    [31.8170]    [176.7484]
    '剩余'        [     2.1602]    [    12]    [ 0.1800 ]    [         ]
    '总和'        [    33.9771]    [    13]    [         ]    [         ]
Columns 6 through 7
    'Fa 值'    '显著性'
    [4.7472]    '高度显著'
    [9.3302]    [         ]
    [         ]    [         ]
r2 =
     0.9364

```

```
Sy =
    0.4243
yy1 =
    2.9531
    3.0263
    3.0994
    3.1725
    3.2457
    3.3188
```

从运行的结果可以看出，得到的自回归方程是 $y = -2.2393 + 0.7313x$ ；从方差分析表可以看出，回归的效果是高度显著的；从可决系数 $r^2 = 0.9364$ ，非常接近 1，也说明回归的效果很好。yy1 中的数据就是设备能力从 7.1kW/人~7.6kW/人，其劳动生产率的估计值。运行还绘制出了回归线图，如图 8-8 所示。

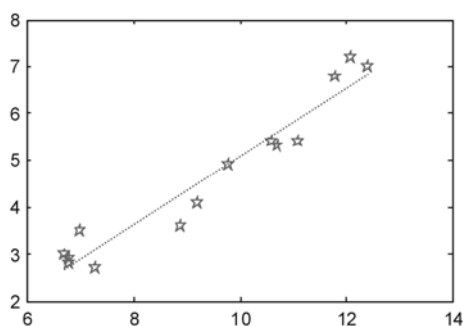


图 8-8 回归线图

接上面的运行结果，再输入以下代码：

```
>> xx2=9.3;
>> pp=0.95;
>> [yy,ylr]=yuce(x,y,xx2,pp)
```

运行程序，输出如下：

```
yy =
    4.5620
ylr =
    4.5210    4.6031
```

这个结果是说，如果建一个设备能力为 9.3kW/人的企业，其劳动生产率的估计值是 4.5620，劳动生产率的 95%置信区间为 (4.5210 4.6031)。

8.3 多元线性回归分析

一元线性回归将影响变量的自变量限制为一个，这在现实的大多社会经济现象中并不易做到，因而应用回归分析时，常需要有更一般的模型，把两个或更多个解释变量的影响分别估计在内。这就是多元回归；亦称多重回归或复回归。当影响因素与因变量之间是线性关系时，所进行的回归分析就是多元线性回归。

8.3.1 多元线性回归模型及矩阵表示

设 y 是一个可观测的随机变量, 它受到 $m-1$ 个非随机因素 x_1, x_2, \dots, x_{m-1} 和随机因素 ε 的影响。若 y 与 x_1, x_2, \dots, x_{m-1} 有如下线性关系:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{m-1} x_{m-1} + \varepsilon \quad (8-20)$$

式 (8-20) 中 $\beta_0, \beta_1, \beta_2, \dots, \beta_{m-1}$ 是未知参数; ε 是均值为零, 方差为 $\sigma^2 > 0$ 的不可观测的随机变量, 称为误差项, 并通常假定 $\varepsilon \sim N(0, \sigma^2)$ 。模型称为多元回归模型, 且称 y 为因变量, x_1, x_2, \dots, x_{m-1} 为自变量。

要建立多元线性回归模型, 首先要估计未知参数 $\beta_0, \beta_1, \beta_2, \dots, \beta_{m-1}$, 为此进行 $n(n \geq p)$ 次独立观测, 得到 n 组数据 (称为样本), 即

$$(x_{i1}, x_{i2}, \dots, x_{im-1}; y_i) \quad (i = 1, 2, \dots, n)$$

它们应满足式 (8-20), 即有

$$\begin{cases} y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \dots + \beta_{m-1} x_{1m-1} + \varepsilon_1 \\ y_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \dots + \beta_{m-1} x_{2m-1} + \varepsilon_2 \\ \vdots \\ y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots + \beta_{m-1} x_{nm-1} + \varepsilon_n \end{cases} \quad (8-21)$$

其中 $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ 相互独立, 且服从 $N(0, \sigma^2)$ 分布。

令

$$\mathbf{Y} = (y_1, y_2, \dots, y_n)^T, \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1m-1} \\ 1 & x_{21} & \dots & x_{2m-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nm-1} \end{bmatrix}, \boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \dots, \beta_{m-1})^T, \boldsymbol{\varepsilon} = (\varepsilon_0, \varepsilon_1, \dots, \varepsilon_n)^T$$

则式 (8-21) 可简写为

$$\begin{cases} \mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} \sim N(0, \sigma^2 \mathbf{I}_n) \end{cases} \quad (8-22)$$

式中: \mathbf{Y} 为观测向量; \mathbf{X} 为设计矩阵, 它们是由观测数据得到的, 是已知的, 并假定 \mathbf{X} 为列满秩的, 即 $r(\mathbf{X})=m$; $\boldsymbol{\beta}$ 是待估计的未知参数向量; $\boldsymbol{\varepsilon}$ 是不可观测的随机误差向量。式 (8-22) 称为多元线性回归模型的矩阵形式, 亦称为高斯-马尔科夫线性模型, 并简记为 $(\mathbf{Y}, \mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n)$ 。

对线性模型 $(\mathbf{Y}, \mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n)$ 所要考虑的问题主要是:

- (1) 估计 $\boldsymbol{\beta}$ 与 σ^2 , 从而建立 y 与 x_1, x_2, \dots, x_{m-1} 之间的关系式。
- (2) 对线性模型假设与 $\boldsymbol{\beta}$ 的某种假设进行检验。
- (3) 对 y 进行预测与对自变量进行控制。

注意: 假定 $n > m$ 。

8.3.2 多元线性回归的显著性检验

1. 相关系数

和一元线性回归分析类似，多元回归也可以用一个“相关系数” R 来衡量，即用回归平方和 SSR 在总平方和 SST 中的比例来衡量，用 R 代替 r ，即有

$$R = \sqrt{\frac{SSR}{SST}}$$

R 称为相关系数，它的意义和一元的相关系数 r 定义一样， $0 \leq R \leq 1$ 。

回归方程的精度用剩余标准差来表示，即

$$S = \sqrt{MSE} = \sqrt{\frac{SSE}{n-m}}$$

2. 方差分析

记 $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ ，则数据的总的离差平方和

$$SST = \sum_{i=1}^n (y_i - \bar{y})^2 \quad (8-23)$$

可分解为两个部分，即

$$SSE = \sum_{i=1}^n (y_i - \hat{y})^2$$

$$SSR = \sum_{i=1}^n (\hat{y} - \bar{y})^2$$

满足

$$SST = SSR + SSE \quad (8-24)$$

对应于 SST 的分解，其自由度也有相应的分解。这里的自由度是指平方和中独立变化项的数目。可以证明， SST 的自由度为 $n-1$ ； SSE 的自由度为 $n-m$ ， SSR 的自由度为 $m-1$ ，因此可建立的方差分析表见表 8-6。

表 8-6 方差分析表

方差来源	偏差平方和	自由度 f	方差	F 值	F_α	显著性
回归	SSR	$m-1$	$MSR = \frac{SSR}{m-1}$	$F = \frac{MSR}{MSE}$		
误差	SSE	$n-m$	$MSE = \frac{SSE}{n-m}$			
总和	SST	$n-1$				

为检验 y 与 x_1, x_2, \dots, x_{m-1} 之间是否存在显著的线性回归关系，即检验假设

$$\begin{cases} H_0: b_1=b_2=\dots=b_{m-2}=0 \\ H_1: \text{至少有某个 } b_i \neq 0, 1 \leq i \leq m-1 \end{cases}$$

这是因为若 H_0 成立，则 $\hat{y} = b_0$ ，即 y 与 x_1, x_2, \dots, x_{m-1} 之间不存在线性回归关系，基于上述方差分析表，构造如下检验统计量：

$$F = \frac{\text{MSE}}{\text{MSE}}$$

当 H_0 为真时, 可以证明 $F \sim F(m-1, n-m)$, 且若 H_0 不真, F 的值有偏大的趋势。因此, 给定显著性水平 α , 查 F 分布表得临界值 $F_\alpha(m-1, n-m)$, 计算 F 的观测值 F_0 , 若 $F_0 < F_\alpha(m-1, n-m)$, 接受 H_0 , 即在显著性水平 α 之下, 认为线性关系不显著; 若 $F_0 \geq F_\alpha(m-1, n-m)$, 拒绝 H_0 , 即认为 y 与 x_1, x_2, \dots, x_{m-1} 之间存在显著的线性关系。一般地, 取 $\alpha=0.05$ 时, 拒绝 H_0 , 即认为 y 与 x_1, x_2, \dots, x_{m-1} 之间存在显著的线性回归关系, 在方差分析表中的“显著性”一栏中填写“显著”; 取 $\alpha=0.01$ 时, 拒绝 H_0 , 即认为 y 与 x_1, x_2, \dots, x_{m-1} 之间的线性回归关系为高度显著, 在方差分析表中的“显著性”一栏中填写“高度显著”; 否则, 填写“不显著”。

3. 偏回归系数检验

回归关系显著并不意味着每个自变量 $x_i (1 \leq i \leq m-1)$ 对 y 的影响都显著, 可能其中的某个或某些对 y 的影响不显著。一般说来, 总希望从回归方程中剔除那些对 y 的影响不显著的自变量, 从而建立一个较为简单有效的回归方程, 以便于实际应用。因为当一个回归方程包含有不显著变量时, 它不仅对利用回归方程作预测和控制带来麻烦, 而且还会增大 \hat{y} 的方差, 从而影响预测的精度。为此就需要对每一个回归系数作显著性检验, 显然, 若某个自变量 x_i 对 y 无影响, 那么在线性模型中, 它的系数 b_j 应为零。因此, 检验 x_i 的影响是否显著等价于检验假设

$$H_0: b_j = 0; \quad H_1: b_j \neq 0$$

可以证明

$$\frac{\hat{b}_j - b_j}{S(\hat{b}_j)} \sim t(n-m) \quad (j=1, 2, \dots, m-1) \quad (8-25)$$

其中 $S(\hat{b}_j)$ 为 $S(\hat{b}_j) = \text{MSE}(\mathbf{X}^T \mathbf{X})^{-1}$ 的主对角线上的第 j 个元素的平方根。由此可知, 若 H_0 为真时, 有

$$t = \frac{\hat{b}_j}{S(\hat{b}_j)} \sim t(n-m) \quad (8-26)$$

若 H_0 不真, 则 t 有偏大趋势。在显著水平 α 下, 查表得 $t_{\frac{\alpha}{2}}(n-m)$ 。记 t 的观测值为 t_0 , 检验准则为

$$\begin{cases} \text{若 } |t_0| < t_{\frac{\alpha}{2}}(n-m), \text{ 则接受 } H_0 \\ \text{若 } |t_0| \geq t_{\frac{\alpha}{2}}(n-m), \text{ 则拒绝 } H_0 \end{cases} \quad (8-27)$$

另外, 还可求得 b_j 的置信度 $1-\alpha$ 的置信区间为

$$\hat{b}_j \pm t_{\frac{\alpha}{2}}(n-m)S(\hat{b}_j) \quad (8-28)$$

8.3.3 β 的最小二乘估计

这一节讨论线性模型 $(Y, X\beta, \sigma^2 I_n)$ 中未知参数 $\beta_0, \beta_1, \beta_2, \dots, \beta_{m-1}$ 和 σ^2 的点估计, 所用的方法仍是最小二乘法。

设

$$Q = \varepsilon^T \varepsilon = (Y - X\beta)^T (Y - X\beta) \quad (8-29)$$

即

$$Q = \sum_{i=1}^n \varepsilon_i^2 = \sum_{i=1}^n \left(y_i - \sum_{j=0}^{m-1} x_{ij} \beta_j \right)^2 \quad (8-30)$$

其中 $x_{i0} = 1 (i=1, 2, \dots, n)$ 。称 Q 为误差平方和, Q 反映了 y 与 $\sum_{j=0}^{m-1} x_{ij} \beta_j$ (这里初始值 $x_{i0} = 1$) 之间在 n 次观察中总的误差程度, Q 越小越好, 由式 (8-29) 知 Q 是未知参数向量 β 的非负二次函数, 因此, 可取使得 Q 达到最小值时 β 的值 $\hat{\beta}$ 作为 β 的点估计。因此 $\hat{\beta}$ 应满足如下关系:

$$(Y - X\hat{\beta})^T (Y - X\hat{\beta}) = \min_{\beta} \{ (Y - X\beta)^T (Y - X\beta) \} \quad (8-31)$$

即

$$\sum_{i=1}^n \left(y_i - \sum_{j=0}^{m-1} x_{ij} \beta_j \right)^2 = \min_{\beta_0, \beta_1, \dots, \beta_{m-1}} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=0}^{m-1} x_{ij} \beta_j \right)^2 \right\} \quad (8-32)$$

为了求 $\hat{\beta}$, 将式 (8-30) 的 Q 对 β 求导, 并令其为零, 即

$$\frac{dQ}{d\beta} = 0$$

称上式为正规方程。因为

$$Q = (Y - X\beta)^T (Y - X\beta) = Y^T Y - \beta^T X^T Y - Y^T X \beta + \beta^T X^T X \beta = Y^T Y - 2\beta^T X^T Y + \beta^T X^T X \beta$$

又因

$$\frac{d}{d\beta} (\beta^T X^T Y) = X^T Y \quad (8-33)$$

$$\frac{d}{d\beta} (\beta^T X^T X \beta) = 2X^T X \beta \quad (8-34)$$

所以得正规方程

$$X^T X \beta = X^T Y \quad (8-35)$$

8.3.4 误差方差 σ^2 的估计

将自变量的各组观测值代入回归方程, 可得因变量的各估计值 (称为拟合值) 为

$$\hat{Y} \triangleq (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n) = X\hat{\beta}$$

称

$$e \triangleq Y - \hat{Y} = Y - X\hat{\beta} = [I - X(X^T X)^{-1} X^T] Y = (I - H) Y \quad (8-36)$$

为残差向量或剩余向量, 其中 $H = X(X^T X)^{-1} X^T$ 为 n 阶幂等矩阵, I 为 n 阶单位阵, 称数

$$Q_e = e^T e = (Y - X\hat{\beta})^T (Y - X\hat{\beta}) = Y^T (I - H) Y = Y^T Y - \hat{\beta}^T X^T Y$$

为剩余平方和。

由于 $E(Y) = X\beta$ 且 $(I - H)Y = 0$, 则

$$Q_e = e^T e = (Y - E(Y))^T (I - H)(Y - E(Y)) = \varepsilon^T (I - H) \varepsilon$$

由此可得

$$\begin{aligned} E(e^T e) &= E(\text{tr}(\varepsilon^T (I - H) \varepsilon)) = \text{tr}((I - H)E(\varepsilon \varepsilon^T)) = \sigma^2 \text{tr}(I - X(X^T X)^{-1} X^T) \\ &= \sigma^2 (n - \text{tr}((X^T X)^{-1} X^T X)) = \sigma^2 (n - m) \end{aligned}$$

其中 $\text{tr}(\cdot)$ 表示矩阵的迹。从而

$$\hat{\sigma}^2 \triangleq \frac{1}{n - m} e^T e \quad (8-37)$$

为 σ^2 的一个无偏估计。

8.3.5 多元线性回归的预测

在多元线性回归分析中, 当回归方差 $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$ 具有统计显著性时, 利用回归方差容易实现对因变量 y 的预测, 其方法同一元的情形, 这里仅作扼要介绍。

设预测点为 $\mathbf{x}_0 = (x_{01}, x_{02}, \cdots, x_{0p})^T$, 则

$$\hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_{01} + \hat{\beta}_2 x_{02} + \cdots + \hat{\beta}_p x_{0p} \quad (8-38)$$

是对

$$E(y_0) = \beta_0 + \beta_1 x_{01} + \beta_2 x_{02} + \cdots + \beta_p x_{0p} \quad (8-39)$$

的点估计, 也是对

$$y_0 = \beta_0 + \beta_1 x_{01} + \beta_2 x_{02} + \cdots + \beta_p x_{0p} + \varepsilon_0 \quad (\varepsilon_0 \sim N(0, \sigma^2)) \quad (8-40)$$

的点预测。并且, 可以证明统计量

$$t = \frac{y_0 - \hat{y}_0}{\hat{\sigma}^* \Delta} \sim t(n - p - 1) \quad (8-41)$$

其中

$$\hat{\sigma}^{*2} = \frac{\text{SSE}}{n - p - 1}, \quad \Delta = \sqrt{1 + \frac{1}{n} + \sum_{i=1}^p \sum_{j=1}^p (x_{0i} - \bar{x}_i)(x_{0j} - \bar{x}_j) c_{ij}} \quad (8-42)$$

$$\bar{x}_i = \frac{1}{n} \sum_{k=1}^n x_{ki} \quad (i = 1, 2, \cdots, p)$$

于是, 点预测的边际误差为 $\pm t_{1-\frac{\alpha}{2}}(n - p - 1) \hat{\sigma}^* \Delta$, 即在 x_0 处的区间预测为

$$\left(\hat{y}_0 - t_{1-\frac{\alpha}{2}}(n - p - 1) \hat{\sigma}^* \Delta, \hat{y}_0 + t_{1-\frac{\alpha}{2}}(n - p - 1) \hat{\sigma}^* \Delta \right) \quad (8-43)$$

即

$$P\left\{\hat{y}_0 - t_{1-\frac{\alpha}{2}}(n-p-1)\hat{\sigma}^* \Delta < y_0 < \hat{y}_0 + t_{1-\frac{\alpha}{2}}(n-p-1)\hat{\sigma}^* \Delta\right\} \geq 1-\alpha \quad (8-44)$$

当 n 较大, $x_{0i} \approx \bar{x}_i$ ($i=1,2,\dots,p$) 时, 可取 $\Delta=1$ 来简化计算。

8.3.6 多元线性回归的实现

在 MATLAB 中, 用函数 `regress` 还可以进行多元线性回归分析, 其调用格式如下:

`b = regress(y,X)`: 根据输入参数 y 和 X , 用最小二乘法求线性回归系数 b 。

`[b,bint,r,rint,stats] = regress(y,X)`: 返回参数 b 的 95% 置信区间 `bint`; 返回残差 r 向量以及残差向量的 95% 置信区间 `rint`; 返回值 `stats` 是一个有三个分量的向量, 其中第一个分量是可决系数 r^2 、第二个分量是 F 比、第三个分量是回归的 p 值。

`[...] = regress(y,X,alpha)`: 设置输入参数 α 。可以求置信度是 $100 \times (1-\alpha)\%$ 的置信区间。

一般的多元线性回归方程是 $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{m-1} x_{m-1}$ 形式, 在使用 `regress` 函数进行多元线性回归分析时, 由于 `regress` 函数是根据输入参数 X 与 y 求回归系数, 这就需要使用者在输入参数 X 的第一列创建一个全 1 列。也就是说, 如果 X 的每一列是一个自变量的观测值, 为了在回归方程中有常数, 应该把输入参数变成 $X=[\text{ones}(\text{length}(X), 1), X]$ 形式。

由于用函数 `regress` 进行多元线性回归分析, 必须在其输入参数 X 的第一列补上一列全 1 列, 且输出参数中没有方差分析表, 可以通过编程来解决这些问题。

1. 回归系数

为了使用方便, 可以用下面的函数求出多元回归方程的系数:

```
function beta=hgxshu2(X,y)
if nargin~=2
    error('多元线性回归要求有两个输入参数');
end
[n,p]=size(X);
[n1,collhs]=size(y);
if n~=n1,
    error('输入参数 y 的行数,必须等于输入参数 X 的行数. ');
end
if collhs~=1,
    error('输入参数 y 应该是一个列向量');
end
A=[ones(n,1),X];
beta=A\y;
函数的输出参数就是回归方程的系数。
```

2. 多元线性回归方程的检验

对多元线性回归可以用方差分析、可决系数或标准差来检验拟合程度的好坏, 其调用函数如下:

```
function [table,r2,Sy,T,TT]=dyxxhgdjyan(X,y,alpha,alpha3)
```



```

if nargin<3
    alpha=[0.05,0.1];
end
if nargin<4
    alpha3=0.05;
end
[n,p]=size(X);
if n<p
    error('观测值的数目过少');
end
m=p+1;
beta=hgxshu2(X,y); % 计算回归系数
A=[ones(n,1),X];
yhat=A*beta;
SSR=(yhat-mean(y))*(yhat-mean(y));
SSE=(yhat-y)*(yhat-y);
SST=(y-mean(y))*(y-mean(y));
Fb=SSR/(m-1)/SSE*(n-m);
Falpha=finv(1-alpha,m-1,n-m);
table=cell(4,7);
table(1,:)={'方差来源','偏差平方和','自由度','方差','F值','Fa值','显著性'};
table(2,1:6)={'回归',SSR,m-1,SSR/(m-1),Fb,min(Falpha)};
table(3,1:6)={'剩余',SSE,n-m,SSE/(n-m),[],max(Falpha)};
table(4,1:3)={'总和',SST,n-1};
if Fb>=max(Falpha)
    table{2,7}='高度显著';
elseif (Fb<max(Falpha))&(Fb>=min(Falpha))
    table{2,7}='显著';
else
    table{2,7}='不显著';
end
r2=SSR./SST;
Sy=sqrt(SSE/(n-m));
SB=SSE/(n-m)*inv(A'*A);
Sb=sqrt(diag(SB));
t=ttinv(1-alpha3/2,n-m);
T=(beta./Sb)>t;
TT=[beta-t*Sb,beta+t*Sb];
T(1)=[];
TT(1,:)=[];

```

这一函数可以进行多元线性回归的方差分析，计算可决系数和剩余标准误差，以及各自变量的显著性、回归系数的置信区间。

输入参数 X 是 n 行 p 列的矩阵，每一列是一个自变量的观测值；每一行是自变量的一组观测值。要求矩阵的行数不能小于矩阵的列数。

输入参数 α 为了判断方差分析表中回归的显著水平是“高度显著”，还是“显著”。 α 应该有两个值，且取值范围必须为 $(0,1)$ 。默认值 $\alpha=[0.05, 0.01]$ ，即 0.05 为显著水平；0.01 为高度显著水平。

α_3 是对各自变量的回归显著性进行判断时的显著水平，默认值为 0.05。

输出参数 table 是方差分析表； r^2 是可决系数 r^2 ； S_y 是剩余标准误差 S_y ； T 是一个由 0, 1 组成的列向量；1 表示自变量的影响显著；0 表示影响不显著； TT 是一个 p 行两列的矩阵，每一行表示参数置信度为 $1-\alpha_3$ 的置信区间。

3. 预测

用下面的程序可以根据输入数据，进行多线性回归的预测：

```
function [ynew,ylr]=yuce2(X,y,Xnew,pp)
ynew=[];
ylr=[];
if length(Xnew)==0
    return;
end
if (nargin<4)||(~isnumeric(pp))||(pp<=0)||(pp>=1)
    pp=0.95;
end
beta=hgxshu2(X,y);    %调用函数,计算回归系数
[n1,p1]=size(Xnew);
[n,p]=size(X);
if p1~=p
    disp('预测自变量的个数不正确');
    return;
end
[table,r2,Sy,T,TT]=dyxxhgdjyan(X,y);
Xnew=[1;Xnew(:)];
ynew=Xnew'*beta;    %点预测
Ve=Sy*Sy;
X=[ones(n,1),X];
S2=Ve*(1+Xnew'*inv(X'*X)*Xnew);
Sy=sqrt(S2);
ta=tinv(0.5+pp/2,n-p-1);
yl=ynew-ta.*Sy;    %预测区间的左端点
yr=ynew+ta.*Sy;    %预测区间的右端点
ylr=[ yl (:),yr(:)];    %区间预测
```

输入参数 X , y 是原始数据; 输入参数 X_{new} 是预测的自变量; 输入参数 pp 是预测的置信度; 两个输出参数, 第一个输出参数 y_{new} 是预测值; 第二个输出参数 ylr 是预测区间。

4. 总体分析

利用下面的函数可以对多元线性回归进行总体分析:

```
function [beta,table,r2,Sy,T,TT,ynew,ylr]=regres2(X,y,Xnew,pp,alpha,alpha3)
if nargin<3
    Xnew=[];
end
if nargin<4
    pp=0.95;
end
if nargin<5
    alpha=[0.05,0.01];
end
if nargin<6
    alpha3=0.05;
end
[n,p]=size(X);
if n<p
    error('观测值的数目过少');
end
beta=hgxshu2(X,y); % 计算系数
[table,r2,Sy,T,TT]=dyxxhgdjyan(X,y,alpha,alpha3);
[ynew,ylr]=yuce2(X,y,Xnew,pp);
```

函数 `regres2` 通过调用上面已经创建的各个函数, 对多元线性回归进行总体分析。其中输入参数 X 、 y 是原始数据; X_{new} 可选, 是预测的自变量; pp 可选, 是预测的置信度, 默认值是 0.95; α 可选, 用来决定方差分析表中的显著性为显著或高度显著, 默认值 $\alpha=[0.05, 0.01]$; α_3 可选, 用来确定各自变量显著性的显著水平, 默认值是 0.05。输出参数 β 是多元线性回归的系数, 是一个列向量; table 是方差分析表; r^2 是可决系数 r^2 ; S_y 是标准误差 S_y ; T 是各自变量的显著性; TT 是回归系数的置信区间; y_{new} 是点预测值, 如果没有输入 X_{new} 时, $y_{\text{new}}=[]$; ylr 是预测区间, 如果没有输入 X_{new} 时, $ylr=[]$ 。

【例 8-11】铝合金化学铣切工艺中, 为了便于生产操作, 需要对腐蚀速度进行控制, 因此要考虑腐蚀液温度、腐蚀液含铝量、碱浓度对腐蚀速度的影响, 一共做了 44 次试验, 所得数据见表 8-7。试分析腐蚀速度对三个变量的回归方程, 并预测腐蚀液温度为 90°C 、腐蚀液含铝量为 90.10g/L 、碱浓度为 260g/L 的腐蚀速度的置信度 95% 的置信区间。

表 8-7 铝合金化学铣切工艺试验数据表

试验号	温度 $x_1/^{\circ}\text{C}$	含铝量 $x_2/(\text{g/L})$	碱浓度 $x_3/(\text{g/L})$	腐蚀速 度 y	试验号	温度 $x_1/^{\circ}\text{C}$	含铝量 $x_2/(\text{g/L})$	碱浓度 $x_3/(\text{g/L})$	腐蚀速 度 y
1	73	12	200	0.0240	23	87	48	200	0.0325
2	73	21	200	0.0235	24	87	19	200	0.0254
3	75	30	200	0.0240	25	77	19	150	0.0230
4	75	36	200	0.0245	26	77	19	175	0.0254
5	75	42	200	0.0190	27	77	19	200	0.0263
6	75	48	200	0.0185	28	77	19	225	0.0312
7	79	12	200	0.032	29	77	19	250	0.0329
8	79	21	200	0.030	30	81	27	150	0.0241
9	79	30	200	0.029	31	81	27	175	0.0238
10	79	36	200	0.0275	32	81	27	200	0.0361
11	79	42	200	0.0250	33	81	27	225	0.0216
12	79	48	200	0.0225	34	81	27	250	0.0321
13	83	12	200	0.0375	35	85	35	150	0.0245
14	83	21	200	0.0562	36	85	36	175	0.0365
15	83	30	200	0.0123	37	85	24	200	0.0341
16	83	36	200	0.0243	38	85	42	225	0.0234
17	83	42	200	0.0362	39	85	39	250	0.0298
18	83	48	200	0.0254	40	89	36	150	0.0387
19	87	12	200	0.0241	41	89	46	175	0.0368
20	87	21	200	0.0364	42	89	41	200	0.217
21	87	30	200	0.0502	43	89	39	225	0.0289
22	87	36	200	0.0402	44	89	38	250	0.0329

其实现的 MATLAB 程序代码如下:

```
>> clear all;
X=[73 12 200;73 21 200;75 30 200;75 36 200;75 42 200;75 48 200;79
12 200;79 21 200;79 30 200;79 36 200;79 42 200;...
79 48 200;83 12 200;83 21 200;83 30 200;83 36 200;83 42 200;...
83 48 200;87 12 200;87 21 200;87 30 200;...
87 36 200;87 48 200;87 19 200;77 19 150;77 19 175;77 19 200;...
77 19 225;77 19 250;81 27 150;81 27 175;81 27 200;81 27 225;...
81 27 250;85 35 150;85 36 175;85 24 200;85 42 225;85 39 250;...
89 36 150;89 46 175;89 41 200; 89 39 225;89 38 250];
y=[0.0240;0.0235;0.0240;0.0245;0.0190;0.0185;0.032;0.030;0.029;0.0275;0.0250;0.0225;...
0.0375;0.0562;0.0123;0.0243;0.0362;0.0254;0.0241;0.0364;0.0502;0.0402;0.0325;0.0254;...
0.0230;0.0254;0.0263;0.0312;0.0329;0.0241;0.0238;0.0361;0.0216;0.0321;0.0245;0.0365;...
0.0341;0.0234;0.0298;0.0387;0.0368;0.217;0.0289;0.0329];
```

```
Xnew=[90,90.10,260];
pp=0.95;
[beta,table,r2,Sy,T,TT,ynew,ylr]=regres2(X,y,Xnew,pp)
```

运行程序，输出如下：

```
beta =
    -0.1403
     0.0021
     0.0001
     0.0000

table =
    '方差来源'    '偏差平方和'    '自由度'    '方差'    'F 值'
    '回归'        [    0.0044]    [    3]    [    0.0015]    [1.7741]
    '剩余'        [    0.0328]    [   40]    [8.2055e-004]    [    ]
    '总和'        [    0.0372]    [   43]    [    ]    [    ]
    'Fa 值'        '显著性'
    [2.8387]        '不显著'
    [4.3126]        [    ]
    [    ]        [    ]

r2 =
    0.1174

Sy =
    0.0286

T =
     1
     0
     0

TT =
    0.0001    0.0040
   -0.0008    0.0009
   -0.0003    0.0004

ynew =
    0.0547

ylr =
   -0.0239    0.1334
```

8.4 偏最小二乘回归分析

8.4.1 偏最小二乘回归分析

设有 q 个因变量 y_1, y_2, \dots, y_q 与 p 个自变量 x_1, x_2, \dots, x_p ，为了研究因变量与自变量的统计关系，观测了 n 个样本点，由此分别构成了自变量与因变量的“样本点 \times 变量”型的数据矩阵，记为

$$\mathbf{X} = (x_{ij})_{n \times p} = (x_1, x_2, \dots, x_p) \quad (8-45)$$

和

$$\mathbf{Y} = (Y_{ij})_{n \times q} = (y_1, y_2, \dots, y_q) \quad (8-46)$$

PLS 方法在建模过程中采用了信息综合与筛选技术, 不直接考虑因变量系统 \mathbf{Y} 对自变量系统 \mathbf{X} 的回归模型, 而是从自变量系统 \mathbf{X} 中逐步提取 m 个对自变量系统 \mathbf{X} 和因变量系统 \mathbf{Y} 都具有最佳解释能力的新综合变量 $t_1, t_2, \dots, t_m (m \leq p)$, 也称为主成分。首先建立 y_k 对主成分 t_1, t_2, \dots, t_m 的 MLR 回归方程, 然后还原为 y_k 关于原自变量系统 x_1, x_2, \dots, x_p 的 PLS 回归方程, 其中 $k=1, 2, \dots, q$ 。

8.4.2 偏最小二乘回归方法的算法步骤

首先要进行预备分析, 目的是判断自变量(因变量)是否存在多重相关性, 判断因变量与自变量是否存在相关关系, 进而决定是否采用 PLS 方法建模。具体计算方法是: 记矩阵 $\mathbf{Z}=(\mathbf{X}, \mathbf{Y})$, 求 \mathbf{Z} 的各列数据之间的简单相关系数; 然后, 按下列步骤建立偏最小二乘回归方程。

1. 标准化原始数据

标准化后的数据矩阵记为 $\mathbf{E}_0 = (e_{ij})_{n \times p}$ 和 $\mathbf{F}_0 = (f_{ij})_{n \times q}$, 其中

$$e_{ij} = \frac{x_{ij} - \bar{x}_j}{s_{x_j}} \quad (i=1, 2, \dots, n; j=1, 2, \dots, p) \quad (8-47)$$

$$f_{ij} = \frac{y_{ij} - \bar{y}_j}{s_{y_j}} \quad (i=1, 2, \dots, n; j=1, 2, \dots, q) \quad (8-48)$$

式(8-47)和式(8-48)中, \bar{x}_j 、 \bar{y}_j 分别为矩阵 \mathbf{X} 与 \mathbf{Y} 的第 j 列数据的平均值; s_{x_j} 、 s_{y_j} 分别为矩阵 \mathbf{X} 与 \mathbf{Y} 的第 j 列数据的标准差。

2. 建立回归方程

1) 建立关于主成分的 MLR 回归方程

求出 \mathbf{F}_0 在 t_1, t_2, \dots, t_m 上的 MLR 回归方程, 即

$$\mathbf{F}_0 = t_1 \mathbf{r}_1^T + t_2 \mathbf{r}_2^T + \dots + t_m \mathbf{r}_m^T + \mathbf{F}_m \quad (8-49)$$

2) 变换为关于标准化变量的 PLS 回归方程

将 $t_i = \mathbf{E}_{i-1} \mathbf{w}_i = \mathbf{E}_0 \mathbf{w}_i^* (i=1, 2, \dots, m)$ 代入方程(8-49), 得 \mathbf{F}_0 关于 \mathbf{E}_0 的 PLS 回归方程为

$$\mathbf{F}_0 = \mathbf{E}_0 \mathbf{w}_1^* \mathbf{r}_1^T + \mathbf{E}_0 \mathbf{w}_2^* \mathbf{r}_2^T + \dots + \mathbf{E}_0 \mathbf{w}_m^* \mathbf{r}_m^T + \mathbf{F}_m \quad (8-50)$$

其中: $\mathbf{w}_i^* = \prod_{k=1}^{i-1} (\mathbf{I} - \mathbf{w}_k \mathbf{p}_k') \mathbf{w}_i (i=1, 2, \dots, m)$; \mathbf{I} 为单位矩阵。

3) 还原为关于原始变量的 PLS 回归方程

将方程(8-50)还原成关于原始变量的 PLS 回归方程

$$\hat{y}_k = \left(\bar{y}_k - \sum_{i=1}^p a_{ki} \frac{s_{y_k}}{s_{x_i}} \bar{x}_i \right) + \sum_{i=1}^p a_{ki} \frac{s_{y_k}}{s_{x_i}} x_i \quad (k=1, 2, \dots, q)$$

式中: a_k 是矩阵 $\mathbf{a}_{p \times q} = \sum_{j=1}^m \mathbf{w}_j^* \mathbf{r}_j'$ 的第 k 个列向量; a_{ki} 是 a_k 的第 i 个分量。

3. 主成分提取

1) 第一轮主成分提取

求矩阵 $\mathbf{E}_0^T \mathbf{F}_0 \mathbf{F}_0^T \mathbf{E}_0$ 的最大特征值所对应的单位特征向量 \mathbf{w}_1 , 得自变量的第 1 个主成分

$$t_1 = \mathbf{E}_0 \mathbf{w}_1 \quad (8-51)$$

求矩阵 $\mathbf{E}_0^T \mathbf{F}_0 \mathbf{F}_0^T \mathbf{E}_0$ 的最大特征值所对应的单位特征向量 \mathbf{c}_1 , 得因变量的第 1 个主成分

$$u_1 = \mathbf{F}_0 \mathbf{c}_1 \quad (8-52)$$

求残差矩阵

$$\mathbf{E}_1 = \mathbf{E}_0 - t_1 \mathbf{p}_1^T \quad (8-53)$$

$$\mathbf{F}_1 = \mathbf{F}_0 - t_1 \mathbf{r}_1^T \quad (8-54)$$

式 (8-53) 中, $p_1 = \frac{\mathbf{E}_0^T t_1}{\|t_1\|^2}$; 式 (8-54) 中, $r_1 = \frac{\mathbf{F}_0^T t_1}{\|t_1\|^2}$ 。

在 PLS 方法中, 称 w_1 为模型效应权重, c_1 为因变量权重, p_1 为模型效应载荷量。

2) 新一轮主成分提取

令 $\mathbf{E}_0 = \mathbf{E}_1$, $\mathbf{F}_0 = \mathbf{F}_1$, 回到 1), 对残差矩阵进行新一轮的主成分提取和回归分析。

设第 h 步的计算结果为

$$t_h = \mathbf{E}_{h-1} \mathbf{w}_h \quad (8-55)$$

$$u_h = \mathbf{F}_{h-1} \mathbf{c}_h \quad (8-56)$$

$$\mathbf{E}_h = \mathbf{E}_{h-1} - t_h \mathbf{p}_h^T \quad (8-57)$$

$$\mathbf{F}_h = \mathbf{F}_{h-1} - t_h \mathbf{r}_h^T \quad (8-58)$$

式 (8-55) ~ 式 (8-58) 中, $h=1, 2, \dots, m$ ($m \leq \text{rank}(\mathbf{E}_0)$), $p_h = \frac{\mathbf{E}_{h-1}^T t_h}{\|t_h\|^2}$, $r_h = \frac{\mathbf{F}_{h-1}^T t_h}{\|t_h\|^2}$ 。

3) 主成分提取的终止准则

PLS 方法不需要选用所有的主成分建模, 而是采用截尾的方法, 即仅选择前 m 个主成分 t_1, t_2, \dots, t_m , 就可以得到一个预测性能较好的模型。因此, 在主成分提取的每一轮计算中, 都要对是否得到了足够多的主成分进行判断。

判断准则常用的有交叉有效性准则和复测定系数准则。

称

$$Q_h^2 = 1 - \frac{\text{PRESS}_h}{\text{SS}_{(h-1)}}$$

为主成分 t_h 关于因变量系统 \mathbf{Y} 的交叉有效性。

上式中各参数的意义如下: PRESS_h 是从所有 n 个样本点中舍弃某个样本点 $x^{(i)} (i=1, 2, \dots, n)$ 之后, 用剩余的 $n-1$ 个样本点拟合出含 h 个主成分的回归方程, 再对 $x^{(i)} (i=1, 2, \dots, n)$ 点进行预测的预测误差平方和。更详细一些, 记 $\hat{y}_{hj}(-i)$ 为 y_j 在样本点 $x^{(i)}$ 上的预测值,

$\text{PRESS}_{hj} = \sum_{i=1}^n [y_{ij} - \hat{y}_{hj(-i)}]^2$ 为 y_j 的预测误差平方和, 则 $\text{PRESS}_h = \sum_{i=1}^p \text{PRESS}_{hi}$ 就是 Y 的预测误差平方和。

$\text{SS}_{(h-1)}$ 是用所有 n 个样本点拟合出的含 $h-1$ 个主成分的回归方程的拟合误差平方和。更详细一点, 记 $\hat{y}_{(h-1)ji}$ 为 y_j 在样本点 $x^{(i)}$ 上的拟合值 $\text{SS}_{(h-1)j} = \sum_{i=1}^n [y_{ij} - \hat{y}_{(h-1)ji}]^2$ 为 y_j 的拟合误差平方和, 则 $\text{SS}_{(h-1)} = \sum_{i=1}^p \text{SS}_{(h-1)i}$ 就是 Y 的拟合误差平方和。

交叉有效性是对新增主成分能否对模型的预测功能有显著改进的判断指标。

若 $Q_h^2 \geq 1 - 0.95^2 = 0.0975$, 则认为主成分 t_h 的边际贡献是显著的。

称

$$Q_h^2 = \frac{\sum_{k=1}^h (\|t_k\|^2 \times \|p_k\|^2)}{\|E_0\|^2}$$

为自变量系统 X 被提取的变异信息量。称

$$R_h^2 = \frac{\sum_{k=1}^h (\|t_k\|^2 \times \|r_k\|^2)}{\|F_0\|^2}$$

为回归方程的复测定系数。

复测定系数表示所提取的主成分的可解释变异信息占总变异的百分比。

当 $h=m$, 复测定系数 R_m^2 的值足够大时, 可在第 m 步终止主成分的提取计算。通常 $R_m^2 \geq 0.85$ 即可。

8.4.3 偏最小二乘回归方法分析

PLS 方法除了前述建模技术, 还包括 PLS 辅助分析技术, 可以在获得一个更为合理的回归模型的同时, 完成一些类似于主成分分析和典型相关分析的研究内容, 提供更加丰富、深入的系统信息。

1. 自变量和因变量之间的相关关系分析

在一元回归分析中, 为了判定自变量和因变量之间的关系, 经常采用散点图来作直观的分析, 简单而有效。这种方法在多元回归分析中遇到困难: 多维数据构成了一个超平面, 难以作直观观察; 各自变量间相互关联, 不能将变量简单地分割开来分析。

PLS 方法的 t_1-u_1 平面图功能使这一点成为可能。

在 PLS 方法中, 自变量集合 X 和因变量集合 Y 之间的相关关系可以通过 t_1 和 u_1 的相关关系得到反映。因此, 绘制以 t_1 为横坐标, u_1 为纵坐标的 t_1-u_1 平面图, 绘出第一主成分偶对 (t_1, u_1) 的观测样本散点图。如果所有样本点 $(t_1(i), u_1(i))$ ($i=1, 2, \dots, n$) 在图中的排列近似于一条直线, 则

说明 X 和 Y 之间存在着较强的相关关系, 这时采用 PLS 方法建立 Y 对 X 的线性模型才是合理的。

2. 自变量对因变量系统的解释能力

PLS 方法中, 自变量对因变量的解释能力是以变量投影重要性指标 (VIP) 来测度的。

自变量对主成分的边际贡献为

$$VIP_j = \sqrt{\frac{p}{Rd(Y; t_1, t_2, \dots, t_m)} \sum_{h=1}^m Rd(Y; t_h) w_{hj}^2}$$

式中: w_{hj} 是主轴 w_h 的第 j 个分量; $Rd(Y; t_h)$ 、 $Rd(Y; t_1, t_2, \dots, t_m)$ 分别是 t_h 对 Y 的解释能力和 t_1, t_2, \dots, t_m 对 Y 的累计解释能力。

VIP_j 定义式的意义是基于这样一个事实: 由于 x_j 对 Y 的解释是通过 t_h 来传递的, 如果 t_h 对 Y 的解释能力很强, 而 x_j 在构造 t_h 时又起到了相当重要的作用, 则 x_j 对 Y 的解释能力就被视为很大。也就是说, 如果在 $Rd(Y; t_h)$ 值很大时的 t_h 成分上, w_{hj} 取得很大的值, 则 x_j 对解释 Y 就有很重要的作用。

另外, 容易证明 $\sum_{j=1}^p VIP_j^2 = p$, 所以, 对于 p 个自变量 x_j ($j=1, 2, \dots, p$), 如果它们在解释 Y

时的作用都相同, 则所有 VIP_j 均等于 1; 否则, 对于 $VIP_j (>1)$ 很大的 x_j , 它在解释因变量 Y 时就有更加重要的作用。

统计工具箱提供了两个主成分分析函数 `princomp` 和 `pcacov`。

1) `princomp` 函数

其调用格式如下:

```
[COEFF, SCORE] = princomp(X)
```

```
[COEFF, SCORE, latent] = princomp(X)
```

```
[COEFF, SCORE, latent, tsquare] = princomp(X)
```

```
[...] = princomp(X, 'econ')
```

其中: X 是 $n \times p$ 的原始数据矩阵; $COEFF$ 为返回主成分的系数, 为 $p \times p$ 矩阵, 每一列为一个主成分的系数; $SCORE$ 为返回原数据在新坐标系中的新数据; $latent$ 为返回 X 协方差矩阵的特征值; $tsquare$ 为返回每个数据点的 Hotelling 统计量。

【例 8-12】`princomp` 函数应用示例。

```
>> load hald;           %MATLAB 自带的数据库
[pc,score,latent,tsquare] = princomp(ingredients);
pc,latent
```

运行程序, 输出如下:

```
pc =
    0.0678    0.6460   -0.5673    0.5062
    0.6785    0.0200    0.5440    0.4933
   -0.0290   -0.7553   -0.4036    0.5156
   -0.7309    0.1085    0.4684    0.4844

latent =
    517.7969
```

67.4964
12.4054
0.2372

2) pcacov 函数

其调用格式如下:

COEFF = pcacov(V)

[COEFF,latent] = pcacov(V)

[COEFF,latent,explained] = pcacov(V)

其中: V 是协方差矩阵; COEFF 为返回主成分; latent 为返回 V 协方差矩阵的特征值。

【例 8-13】pcacov 函数应用示例。

```
>> load hald %MATLAB 自带的数据库
covx = cov(ingredients);
[COEFF,latent,explained] = pcacov(covx)
COEFF =
    -0.0678    0.6460   -0.5673    0.5062
    -0.6785    0.0200    0.5440    0.4933
     0.0290   -0.7553   -0.4036    0.5156
     0.7309    0.1085    0.4684    0.4844
latent =
    517.7969
     67.4964
     12.4054
      0.2372
explained =
     86.5974
     11.2882
      2.0747
      0.0397
```

【例 8-14】根据表 8-8 中人头发的元素的分析结果进行主成分分析。

表 8-8 人头发的元素分析

样本	Cu	Mn	Cl	Br	I
1	9.2	0.30	1770	12.0	3.6
2	12.4	0.39	930	50.0	2.3
3	7.2	0.32	2750	65.3	3.4
4	10.2	0.36	1500	3.4	5.3
5	10.1	0.50	1040	39.2	1.9
6	6.5	0.20	2490	90.0	4.6
7	5.6	0.29	2940	88.0	5.6
8	11.8	0.42	867	43.1	1.5
9	8.5	0.25	1620	5.2	6.2

其实现的 MATLAB 程序代码如下:

```
>> clear all;
x=[9.2,0.30,1770,12.0,3.6;12.4,0.39,930,50.0,2.3;7.2,0.32,2750,65.3,3.4;...
    10.2,0.36,1500,3.4,5.3;10.1,0.50,1040,39.2,1.9;6.5,0.20,2490,90.0,4.6;...
    5.6,0.29,2940,88.0,5.6;11.8,0.42,867,43.1,1.5;8.5,0.25,1620,5.2,6.2];
stdr=std(x);
sr=x./stdr(ones(9,1),:);
[pcs,newdata,variances,t2]=princomp(sr); %主成分分析
pcs          %主成分
newdata      %得分
variances    %方差
t2           %统计量
plot(newdata(:,1),newdata(:,2),'*');
gname        %获取各点代表的样本
```

运行程序输出结果为如下, 效果如图 8-9 所示。

```
pcs =
    -0.5215    0.1028   -0.4127    0.1820   -0.7170
    -0.4652   -0.2691    0.7899    0.2833   -0.0829
     0.5174   -0.1704    0.3127   -0.3823   -0.6778
     0.2769   -0.7610   -0.2824    0.5140   -0.0175
     0.4090    0.5558    0.1680    0.6901   -0.1393
newdata =
    -0.1658    0.7783   -0.0895   -0.6913    0.0216
    -1.8837   -0.4626   -0.6649    0.3083   -0.2388
     1.2205   -0.8723    0.3504   -0.5121   -0.2328
    -0.5357    1.4566    0.3854    0.2579   -0.2436
    -2.0422   -0.7931    0.7886    0.0859    0.3139
     2.3119   -0.6715   -0.7569    0.0573    0.2033
     2.5738   -0.7007    0.4732    0.4217   -0.0676
    -2.1928   -0.6658   -0.3444   -0.0465    0.0418
     0.7139    1.9312   -0.1420    0.1188    0.2022
variances =
     3.3513
     1.1807
     0.2849
     0.1383
     0.0448
t2 =
     4.0149
     4.7531
     4.6267
     4.2103
     6.2156
     4.9348
     4.5668
     2.2815
     4.3964
```

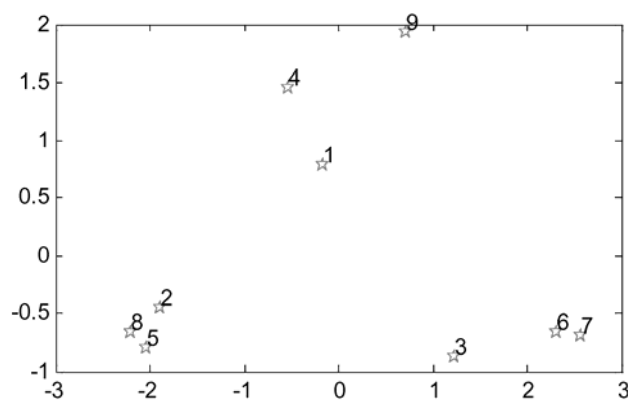


图 8-9 主成分分析

从变量 (Variances) 结果可看出, 共有 5 个主成分, 但前面两个的主成分作用显著, 占了总方差的 90%。

第 9 章 数理统计的其他分析

9.1 聚类分析

聚类分析是一种多元统计分类方法，该法可以对一群不知类别的观察对象按彼此相似的程度进行分类，以达到“物以类聚”的目的。利用聚类分析有助于挑选变量，分析影响因素。在 MATLAB 中提供了一些函数实现聚类分析，分别介绍如下。

9.1.1 MATLAB 实现聚类分析

1. pdist 函数

功能：计算观察量之间的匹配距离。其调用格式如下：

$y = \text{pdist}(X, \text{metric})$ ：使用 metric 指定的方法计算 X 矩阵中配对样本的欧几里得距离。

$y = \text{pdist}(X, 'minkowski', p)$ ：使用明可斯基距离计算对象之间的距离， p 为幂次。

度量欧几里得距离的公式如下：

$$d_{ij} = \sqrt{\sum_{m=1}^M (x_{mi} - x_{mj})^2}$$

【例 9-1】实现欧几里得距离示例。

```
>> X = [1 2; 1 3; 2 2; 3 1]
X =
     1     2
     1     3
     2     2
     3     1
>> Y = pdist(X, 'mahal')
Y =
     2.3452     2.0000     2.3452     1.2247     2.4495     1.2247
>> Y = pdist(X)
Y =
     1.0000     1.0000     2.2361     1.4142     2.8284     1.4142
>> squareform(Y)
ans =
     0         1.0000     1.0000     2.2361
     1.0000     0         1.4142     2.8284
     1.0000     1.4142     0         1.4142
     2.2361     2.8284     1.4142     0
```

2. squareform 函数

功能：将 pdist 函数的输出重定义为平方矩阵的格式。其调用格式如下：

$Z = \text{squareform}(y)$ ：将 pdist 函数返回的距离信息 y 重新定义为平方矩阵的格式。该格式中

$Z(i, j)$ 表示原始数据中 i 观察量和 j 观察量之间的距离。

【例 9-2】squareform 函数应用示例。

```
>> y = 1:6
y =
    1    2    3    4    5    6
>> X = [0 1 2 3; 1 0 4 5; 2 4 0 6; 3 5 6 0]
X =
    0    1    2    3
    1    0    4    5
    2    4    0    6
    3    5    6    0
>> squareform(y)
ans =
    0    1    2    3
    1    0    4    5
    2    4    0    6
    3    5    6    0
```

3. linkage 函数

功能：创建系统聚类树。其调用格式如下：

$Z = \text{linkage}(y)$ ：使用最短距离法创建一个系统聚类树， y 由 pdist 函数生成。

$Z = \text{linkage}(y, \text{method})$ ：用 method 指定的算法计算系统聚类树。

【例 9-3】创建系统聚类树示例。

```
>> X = [3 1.7; 1 1; 2 3; 2 2.5; 1.2 1; 1.1 1.5; 3 1];
Y = pdist(X);
Z = linkage(Y)
Z =
    2.0000    5.0000    0.2000
    3.0000    4.0000    0.5000
    6.0000    8.0000    0.5099
    1.0000    7.0000    0.7000
    9.0000   11.0000    1.2806
   10.0000   12.0000    1.3454
```

4. dendrogram 函数

功能：输出冰注图。其调用格式如下：

$H = \text{dendrogram}(Z)$ ：生成系统聚类树 Z 的冰注图， Z 由 linkage 函数生成。

$H = \text{dendrogram}(Z, p)$ ：生成只有顶部 p 个节点的冰注图。

$[H, T] = \text{dendrogram}(\dots)$ ：创建一个冰注图，返回一个大小为 m 的向量 T ，其中包含了原始数据集中每一个对象的聚类个数。

【例 9-4】输出随机数冰注图示例。

```
>> clear all;
X = rand(100,2);
Y = pdist(X,'cityblock');
Z = linkage(Y,'average');
[H,T] = dendrogram(Z,'colorthreshold','default');
set(H,'LineWidth',2)
```

运行程序，效果如图 9-1 所示。

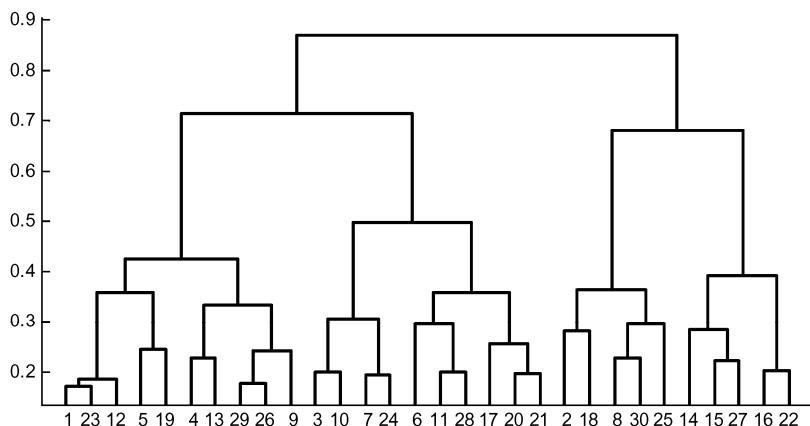


图 9-1 冰注图效果

5. cophenet 函数

功能：计算 Cophenetic 相关系数。其调用格式如下：

`c = cophenet(Z,Y)`: 计算 cophenetic 相关系数，它比较由 `linkage` 函数生成的 `Z` 中的距离信息和由 `pdist` 函数生成的距离信息。

`[c,d] = cophenet(Z,Y)`: `d` 为在同一距离下三角距离为 `y` 的矢量格式。

【例 9-5】计算 Cophenetic 相关系数示例。

```
>> X = [rand(10,3); rand(10,3)+1; rand(10,3)+2];
Y = pdist(X);
Z = linkage(Y,'average');
[c,D] = cophenet(Z,Y);
r = corr(Y,D','type','spearman')
r =
    0.8572
```

6. cluster 函数

功能：根据 `linkage` 函数生成的输出创建聚类。其调用格式如下：

`T = cluster(Z,'cutoff',c)`: 根据 `linkage` 函数生成的系统聚类树 `Z` 来创建聚类。`cutoff` 是一个临界值，它决定 `cluster` 函数怎样聚类。其中 `cutoff` 的值要比 `c` 小。

`T = cluster(Z,'cutoff',c,'depth',d)`: `depth` 指定系统聚类树的水平数，并包含在不连续系统计算中。`d` 为节点的深度。默认值为 2。

【例 9-6】`cluster` 函数应用示例。

```
>> load fisheriris
d = pdist(meas);
Z = linkage(d);
c = cluster(Z,'maxclust',4:5);
crosstab(c(:,1),species)
ans =
     0     0     1
     0    50    47
     0     0     2
```

```

50    0    0
>> crosstab(c(:,2),species)
ans =
    0     4     0
    0    46    47
    0     0     1
    0     0     2
    50     0     0

```

7. clusterdata 函数

功能：根据数据分类。其调用格式如下：

$T = \text{clusterdata}(X, \text{cutoff})$ ：根据 X 创建分类。

【例 9-7】根据数据分类示例。

```

>> rand('state',12);
X = [rand(10,3); rand(10,3)+1.2; rand(10,3)+2.5];
T = clusterdata(X,'maxclust',3);
find(T==3)
scatter3(X(:,1),X(:,2),X(:,3),100,T,'filled')

```

运行程序，输出如下，效果如图 9-2 所示。

```

ans =
    21
    22
    23
    24
    25
    26
    27
    28
    29
    30

```

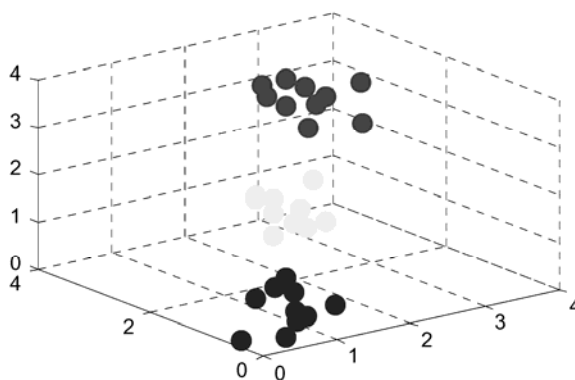


图 9-2 数据分类效果图

8. inconsistent 函数

功能：计算聚类树的不连续系数。其调用格式如下：

$Y = \text{inconsistent}(Z)$ ：对于系统聚类树计算不连续系数。

$Y = \text{inconsistent}(Z, d)$ ：计算系统聚类树中每个联结至深度 d 的不连续系数。

【例 9-8】计算聚类树的不连续系数示例。

```
>> clear all;
rand('state',12);
X = rand(10,2);
Y = pdist(X);
Z = linkage(Y,'single');
dendrogram(Z)
W = inconsistent(Z,3)
```

运行程序，输出如下，效果如图 9-3 所示。

```
W =
    0.1313    0    1.0000    0
    0.1386    0    1.0000    0
    0.1463    0.0109    2.0000    0.7071
    0.2391    0    1.0000    0
    0.1951    0.0568    4.0000    0.9425
    0.2308    0.0543    4.0000    0.9320
    0.2395    0.0748    4.0000    0.7636
    0.2654    0.0945    4.0000    0.9203
    0.3769    0.0950    3.0000    1.1040
```

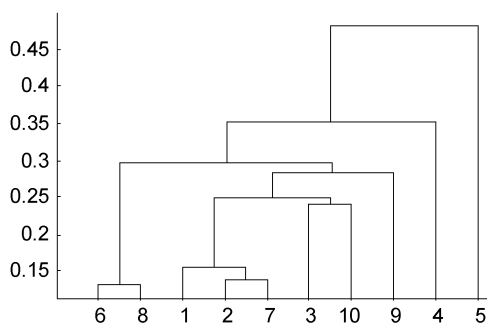


图 9-3 聚类树的不连续系数效果图

对于 MATLAB 内置实现聚类分析的函数已经作简单介绍了，下面通过示例来综合讲述聚类分析在实际中的应用。

【例 9-9】用正辛醇-水分系数、沸点 b.p.、摩尔体积 MV 和分子连接性指数 x 四个参数描述氯苯、1, 4-二氯苯、五氯苯、六氯苯、4-氯硝基苯、硝基苯六个化合物，试根据表 9-1 数据对这六个化合物进行分类。

表 9-1 化合物性质

化合物编号及名称	$\lg K_{ow}$	b.p.	MV	x
氯苯	3.02	131.8	101.9	2.18
1, 4-二氯苯	3.44	247.1	118.2	2.69
五氯苯	5.12	277.1	136.9	4.25
六氯苯	5.41	321.1	138.3	4.78
4-氯硝基苯	2.58	242.0	103.0	2.63
硝基苯	1.87	210.8	102.0	2.11

一种是一次聚类，利用函数可以对样本数据进行一次聚类，但选择面比较窄，不能更改距离的计算方法。

另一种是分布聚类，可以分以下步骤进行分布聚类：① 找到数据集中变量两两之间的相似性和非相似性，用 `pdist` 函数计算变量之间的距离；② 用 `linkage` 函数定义变量之间的连续性；③ 用 `copheneti` 函数评价聚类信息；④ 用 `cluster` 函数创建聚类。

(1) 一次聚类的 MATLAB 代码如下：

```
>> clear all;
X=[3.02 131.8 101.9 2.18;3.44 247.1 118.2 2.69;5.12 277.1 136.9 4.25;...
5.41 321.1 138.3 4.78;2.58 242.0 103.0 2.63;1.87 210.8 102.0 2.11];
T=clusterdata(X,0.5)'
T =
    4    5    2    3    5    1
```

数据集合分为 5 类。调整 `cutoff` 值，将有不同的分类。

(2) 分类聚类 MATLAB 代码如下：

```
>> xx=zscore(X); %数据标准化
Y=pdist(xx); %计算变量间的相似性
squareform(Y); %将输出转化为矩阵,以便阅读
Z=linkage(Y); %定义变量之间的连接
c=cophenet(Z,Y)' %评价聚类信息
c =
    0.8793
```

连接变量生成聚类树后，可以通过下列方法进行修改或了解更多的信息。

① 修改聚类树

衡量聚类信息的有效性用 `cophenet` 函数计算相关性，该值越接近 1，表示聚类效果就越好。

```
>> c=cophenet(Z,Y)
c =
    0.8793
```

将函数中距离计算方法分别指定为“Mahat”、“sEuclid”和“Cityblock”，重新计算 `pdist` 函数后，再用 `cophenet` 计算 `c` 值分别等于 0.5678、0.9358 和 0.9401，所以用“Cityblock”计算距离效果较好。

② 了解与聚类连接相关的更多信息

数据集合中聚类的方法之一是比较聚类树中每一个连接的长度与相邻次一级连接的长度，如果二者相近，则表示此水平上变量之间是相似的，这些连接被认为具有较高水平的连续性；反之，则称为不连续性的。

```
>> dendrogram(Z)
```

生成的聚类树，效果如图 9-4 所示。

聚类树中每一个连接的相对连续可用 `inconsistent` 函数生成的不连续性系数来定量表示。该函数比较某连接的长度与相邻连接长度的均值。若该变量与周围变量连续，则不连续性系数较低，反之则较高。

```
>> I=inconsistent(Z)
```

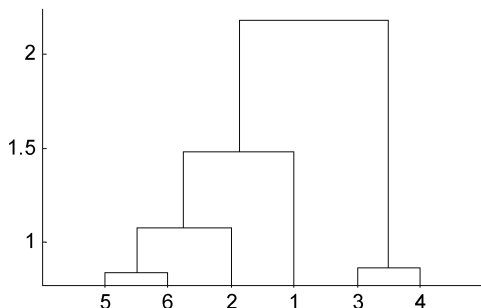


图 9-4 聚类树

I =

0.8399	0	1.0000	0
0.8605	0	1.0000	0
0.9561	0.1644	2.0000	0.7071
1.2753	0.2869	2.0000	0.7071
1.5032	0.6555	3.0000	1.0185

矩阵中，第一列为所有连接长度的均值；第二列为所有连接长度的标准偏差；第三列为计算所包含的连接数；第四列为不连续性系数。该输出信息可以与 `linkage` 函数的输出对照阅读。

```
>> cluster(Z,0.8)' %创建分类，以距离不超过2的不连续性系数为临界点
```

ans =

```
2 2 1 1 2 2
```

从聚类树中可清楚地了解聚类过程。比较起来，化合物3和4的性质与其他化合物相差较大。看来苯环的氢全部或几乎全部被氯取代对化合物的影响是非常显著的。

9.1.2 编程实现聚类分析

9.1.1 节介绍如何从 MATLAB 内置函数实现聚类分析，9.1.2 节，将介绍用户自定义编程实现聚类分析。

聚类分析本质上是将研究的对象进行分类，其基本思想是通过定义样本之间的距离，将相近的样本归为一类，直至所有类之间的距离满足某种条件。聚类分析中常用的是系统聚类法。

系统聚类法的算法过程介绍如下：

- (1) 计算 n 个样本两两之间的距离，形成距离矩阵 D ， D 为对称矩阵，且对角元素为 0。
- (2) 首先构造 n 个类，每一类中只包含一个样本。
- (3) 合并距离最近的两类为新类。
- (4) 计算新类与当前各类的距离，若类的个数已经等于 1，转 (5)，否则转 (3)。
- (5) 决定类的个数和类。

类的距离可以用以下两种算法计算：

- (1) 最短距离法：

$$D(C_1, C_2) = \min_{\substack{x_i \in C_1 \\ x_j \in C_2}} \{d(x_i, x_j)\}$$

- (2) 最长距离法：

$$D(C_1, C_2) = \max_{\substack{x_i \in C_1 \\ x_j \in C_2}} \{d(x_i, x_j)\}$$

其中， $d(x_i, x_j) = \sqrt{\sum_{k=1}^m (x_{ik} - x_{jk})^2}$ 为两个类中任意两个样本的距离。

自定义编程实现的 `JlfenxiAnaly` 函数的源代码如下：

```
function JlfenxiAnaly(X)
format long;
sz = size(X);
N = sz(1); %样本个数
n = sz(2);
D = zeros(n,n);
```

```

totalClass = N;
RecordClass = zeros(N,N+1);
RecordClass(:,1) = ones(N,1);
RecordClass(:,2) = 1:N;
disp('聚类前的 N 个类: ');
disp(RecordClass);

while totalClass > 1
    minClaDist = inf;
    for i=1:totalClass
        for j=i+1:totalClass
            distClass = DpistClass(X,RecordClass,i,j,RecordClass(i,1),RecordClass(j,1));
            if distClass < minClaDist
                minClaDist = distClass;
                i1 = i;
                j1 = j;
            end
        end
    end
    t1 = RecordClass(i1,1);
    t2 = RecordClass(j1,1);
    RecordClass(i1,(t1+2):(t1+t2+1)) = RecordClass(j1,2:(t2+1));
    RecordClass(i1,1) = RecordClass(i1,1) + RecordClass(j1,1);
    RecordClass(j1:(totalClass-1),:) = RecordClass((j1+1):totalClass,:);
    RecordClass(totalClass:N,:) = zeros(N-totalClass+1,N+1);
    totalClass = totalClass - 1;
    str1 = strcat('第',num2str(N - totalClass));
    str1 = strcat(str1, '次聚类, ');
    str1 = strcat(str1, '第');
    str1 = strcat(str1, num2str(i1));
    str1 = strcat(str1, '类和第');
    str1 = strcat(str1, num2str(j1));
    str1 = strcat(str1, '类合并: ');
    disp(str1);
    disp(RecordClass(1:totalClass,:))
end

```

自定义编写的样本距离的计算函数 Dpist 函数的源代码如下:

```

function d = Dpist (X1,X2)
format long;
d = sqrt(dot( X1-X2,X1-X2));
format short;

```

自定义编写的类距离的计算函数 DpistClass 函数的源代码如下:

```

function d = DpistClass(X,C1,l1,l2,n,m)
format long;
d = inf;
for i=1:n
    for j=1:m

```

```

        dc = Dpist(X(C1(l1,i+1),:),X(C1(l2,j+1),:));
        if dc < d
            d = dc;
        end
    end
end
format short;

```

【例 9-10】最短距离算法的系统聚类应用示例。

对下面的样本用最短距离法进行分类。

x1=2 2 3 3 8;

x2=1 4 7 3 0;

其实现的 MATLAB 程序代码如下：

```

>> clear all;
X=[2 1;2 4;3 7;3 3;8 0];
JlfenxiAnaly(X)

```

聚类前的 N 个类：

1	1	0	0	0	0
1	2	0	0	0	0
1	3	0	0	0	0
1	4	0	0	0	0
1	5	0	0	0	0

第 1 次聚类，第 2 类和第 4 类合并：

1	1	0	0	0	0
2	2	4	0	0	0
1	3	0	0	0	0
1	5	0	0	0	0

第 2 次聚类，第 1 类和第 2 类合并：

3	1	2	4	0	0
1	3	0	0	0	0
1	5	0	0	0	0

第 3 次聚类，第 1 类和第 2 类合并：

4	1	2	4	3	0
1	5	0	0	0	0

第 4 次聚类，第 1 类和第 2 类合并：

5	1	2	4	3	5
---	---	---	---	---	---

输出数据的行数代表每次聚类后类的个数，每一行的第一个数表示此类中样本的个数，后面的数代表此类中的样本编号；从程序的输出结果看，聚类的过程一目了然，未聚类前，有 5 个类，每个类各有一个样本；第一次聚类是将第 1 类和第 2 类合并，得到新的第 1 类，此时第 1 类有 2 个样本；而第二次聚类是将第 3 类和第 4 类合并，得到新的第 3 类；第三次聚类是将第 2 类和第 3 类合并，此时总共有 2 个类；经过第四次聚类，类的个数变为 1，聚类的过程至此完成。

9.2 判别分析

判别分析在多元统计分析中也属于数值分类法，但与聚类分析有明显的差别。在判别分析中用以建立判别函数的数据是事先已知所属的类别，而聚类分析的数据类别是未知的。

判别分析在化学、生物学、地质学、石油等领域得到较为广泛的应用。如在化学研究中，可以根据建立的判别函数判别化合物的性质类别。

在 MATLAB 中提供了一些函数实现判别分析，分别介绍如下。

9.2.1 MATLAB 实现判别分析

1. classify 函数

功能：线性判别分析。其调用格式如下：

```
class = classify(sample,training,group)
```

```
class = classify(sample,training,group,type)
```

```
class = classify(sample,training,group,type,prior)
```

```
[class,err] = classify(...)
```

```
[class,err,POSTERIOR] = classify(...)
```

```
[class,err,POSTERIOR,logp] = classify(...)
```

```
[class,err,POSTERIOR,logp,coeff] = classify(...)
```

说明：指定 sample 数据中的每一行到训练集 training 指定的一个类中；group 指明训练集中的每一行属于哪一个类；返回 class，它的每一个元素指定 sample 中对应元素的分类；type 为其判别的类型；err 为返回的误差；POSTERIOR 为返回的概率估计矩阵；logp 为返回一个载体常数；coeff 为返回的结构数组。

【例 9-11】线性判别分析示例。

```
>> load fisheriris           %MATLAB 自带数据
SL = meas(51:end,1);
SW = meas(51:end,2);
group = species(51:end);
h1 = gscatter(SL,SW,group,'rb','p',[,], 'off');
set(h1,'LineWidth',2)
legend('费希尔云芝','费希尔锦葵','位置','NW');
figure;
[X,Y] = meshgrid(linspace(4.5,8),linspace(2,4));
X = X(:); Y = Y(:);
[C,err,P,logp,coeff] = classify([X Y],[SL SW],group,'quadratic');
hold on;
gscatter(X,Y,C,'rb','.',1,'off');
K = coeff(1,2).const;
L = coeff(1,2).linear;
Q = coeff(1,2).quadratic;
f = sprintf('0 = %g+%g*x+%g*y+%g*x^2+%g*x.*y+%g*y.^2,...
            K,L,Q(1,1),Q(1,2)+Q(2,1),Q(2,2));
h2 = ezplot(f,[4.5 8 2 4]);
```

```
set(h2,'Color','m','LineWidth',2)
axis([4.5 8 2 4])
xlabel('萼片长度')
ylabel('萼片宽度')
title('\bf 分类与萼片训练数据');
```

运行程序，输出结果如图 9-5 及图 9-6 所示。

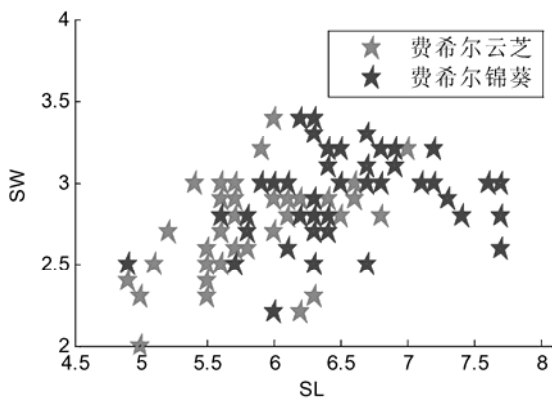


图 9-5 费希尔数据的分类效果

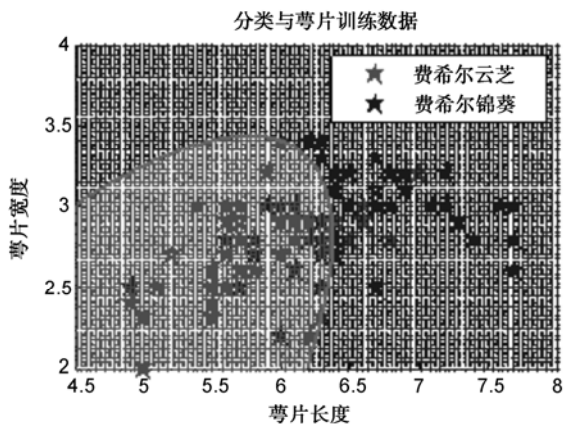


图 9-6 分类与萼片训练数据效果

2. mahal 函数

功能：计算马氏距离。其调用格式如下：

$d = \text{mahal}(Y, X)$ ：计算 X 样本至 Y 中每一个点（行）的马氏距离。

【例 9-12】计算马氏距离示例。

```
>> X = mvnrnd([0;0],[1 .9;.9 1],100);
Y = [1 1;-1 -1;-1 1;-1 -1];
d1 = mahal(Y,X) % 马氏
d1 =
    0.6288
   19.3520
   21.1384
    0.9404
>> d2 = sum((Y-repmat(mean(X),4,1)).^2, 2) % 欧氏
```

```
d2 =
    1.6170
    1.9334
    2.1094
    2.4258
>> scatter(X(:,1),X(:,2))
hold on
scatter(Y(:,1),Y(:,2),100,d1,'p','LineWidth',2)
hb = colorbar;
ylabel(hb,'马氏距离')
legend('X','Y','位置','NW')
```

运行程序，效果如图 9-7 所示。

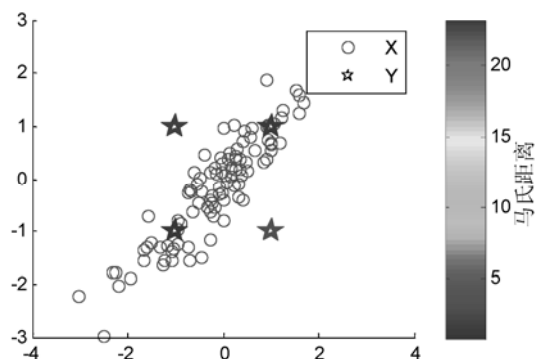


图 9-7 马氏距离计算效果图

对于 MATLAB 内置实现判别分析的函数已经作简单介绍了，下面通过示例来讲述判别分析在实际中的应用。

【例 9-13】以 $\lg(1/EC_{50})$ 1.5 作为活性高低的界限，测定了 26 个含硫芳香族化合物对发光菌的毒性数据。分别计算了这些化合物的 $\lg K_{ow}$ 、Hammett 电荷效应常数 σ ，并测定了水解速率常数 k ，试根据活性类别（两类）及变量 $\lg K_{ow}$ 、 σ 和 $\lg k$ 所取的数据，对三个未知活性同系物的活性进行判别。其数据见表 9-2。

表 9-2 26 个化合物的结构参数与判别结果

化合物编号与类别		$\lg(1/EC_{50})$	σ	$\lg K_{ow}$	pk
1	第 I 类 (低活性)	0.96	1.31	2.30	1.76
2		1.01	0.82	3.20	2.43
3		1.03	0.83	3.41	1.98
4		1.13	1.50	3.80	2.20
5		1.15	1.07	3.86	1.30
6		1.18	1.31	3.07	2.05
7		1.29	1.37	4.31	1.09
8		1.31	1.21	4.33	2.21
9		1.37	1.25	0.95	1.17
10		1.48	1.29	2.31	1.48

续表

化合物编号与类别		$\lg(1/EC_{50})$	σ	$\lg K_{ow}$	pK
11	第 I 类 (低活性)	1.49	1.05	4.27	1.40
12		1.50	1.08	0.99	0.57
13		1.57	1.67	4.32	0.62
14	第 II 类 (高活性)	1.61	1.67	1.89	1.25
15		1.63	1.71	2.29	0.59
16		1.67	1.48	3.00	0.49
17		1.70	1.48	0.96	1.22
18		1.79	1.04	2.27	1.29
19		1.82	1.71	0.66	1.10
20		1.83	1.51	0.95	1.73
21		1.98	2.06	2.27	1.76
22		2.00	1.51	3.00	1.02
23		2.04	2.06	3.00	1.23
24		2.19	1.52	3.01	0.61
25		2.26	1.60	0.77	1.17
26		2.37	2.37	0.98	1.48
27	未知	1.42	0.92	2.79	1.71
28		1.83	1.68	3.35	1.46
29		1.56	1.88	3.11	1.17

其实现的 MATLAB 程序代码如下:

```
>>clear all;
load mydata %保存以上数据为 mydata.mat 文件
training=[a1 a2 a3 a4]
group=[1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2];
sample=[1.33 0.81 2.29 1.71;1.72 1.59 3.35 1.46;1.55 1.71 3.00 1.17];
class=classify(sample,training,group)'
```

运行程序输出为:

```
class = 1 2 2
```

即三个未知化合物的活性类型分别属于低、高、高,与实际结果完全一样。

9.2.2 编程实现判别分析

9.2.1 节介绍如何从 MATLAB 内置函数实现判别分析,这一小节,将介绍用户自定义编程实现判别分析。

判别分析也是一种分类法,但是和聚类分析不同的是,判别分析是在分类已给定的情况下,通过某种判别法则,判断新的样本属于哪个已知类。

在例 9-11 中已经用 MATLAB 的内置函数实现了费希尔判别,下面将编程实现费希尔判别。

费希尔两类判别算法介绍如下:

(1) 输入两类样本值,即

$$\mathbf{X}(A) = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{M1} & x_{M1} & \cdots & x_{Mp} \end{bmatrix}, \mathbf{X}(B) = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{N1} & x_{N1} & \cdots & x_{Np} \end{bmatrix}$$

(2) 计算均值, 即

$$\begin{cases} \bar{X}_l(A) = \frac{1}{M} \sum_{i=1}^M x_{il}(A) \\ \bar{X}_l(B) = \frac{1}{N} \sum_{i=1}^N x_{il}(B) \end{cases} \quad (l=1, 2, \cdots, p)$$

计算方差为

$$S_{jk} = \sum_{i=1}^M [x_{ij}(A) - \bar{X}_j(A)][x_{ik}(A) - \bar{X}_k(A)] + \sum_{i=1}^N [x_{ij}(B) - \bar{X}_j(B)][x_{ik}(B) - \bar{X}_k(B)]$$

($j=1, 2, \cdots, p; k=1, 2, \cdots, p$)

计算距离为

$$d_l = \bar{X}_l(A) - \bar{X}_l(B) \quad (l=1, 2, \cdots, p)$$

(3) 解方程组 $\mathbf{Sc} = \mathbf{d}$ 求出判别系数 c , 建立判别函数

$$Y = \sum_{i=1}^p c_p x_i$$

(4) 用判别函数判别新样本的类别。

自定义编程实现费希尔两类判别法的 Fisherbanbie 函数的源代码如下:

```
function kiX = Fisherbanbie(XA,XB,SaX)
% XA 为第一类的样本矩阵;
% XB 为第二类的样本矩阵;
% SaX 需要分类的样本
% kiX 为样本的类别
format long;
sz1 = size(XA);
sz2 = size(XB);
M = sz1(1);           % 样本个数
N = sz2(1);
n = sz1(2);
meanXA = mean(XA);
meanXB = mean(XB);
sx = zeros(n,n);
Y = zeros(N,n);
for i=1:n
    for j=1:n
        sx(i,j) = dot(XA(:,i)-meanXA(i)*zeros(M,1),XA(:,j)-meanXA(j)*zeros(M,1))+ ...
            dot(XB(:,i)-meanXB(i)*zeros(N,1),XB(:,j)-meanXB(j)*zeros(N,1));
    end
end
```

```

end
d = transpose(meanXA - meanXB);
c = sx\d;
YA = dot(c,meanXA);
YB = dot(c,meanXB);
Yc =(M*YA + N*YB)/(M+N);
Y0 = dot(c,SaX);
if YA > YB
    if Y0 > Yc
        kiX = 1;
        disp('样品属于第一类');
    else
        if Y0 == Yc
            kiX = 0 ;
            disp('没法判断');
        else
            kiX = 2;
            disp('样品属于第二类');
        end
    end
end
else
    if YA < YB
        if Y0 > Yc
            kiX = 2;
            disp('样品属于第二类');
        else
            if Y0 == Yc
                kiX = 0 ;
                disp('没法判断');
            else
                kiX = 1;
                disp('样品属于第一类');
            end
        end
    end
else
    disp('没法判断');
end
end
end

```

【例 9-14】费希尔两类判别法应用示例。其样本数据见表 9-3。

表 9-3 费希尔的样本数据

类别	样本	成分			
		x1	x2	x3	x4
第 I 类	1	13.5	2.79	7.8	49.6
	2	22.31	4.67	12.31	47.8
	3	28.82	4.63	16.18	62.15

续表

类别	样本	成分			
		x1	x2	x3	x4
第 I 类	4	15.29	3.45	7.58	43.2
	5	28.29	4.90	16.12	58.7
第 II 类	1	2.18	1.06	1.22	20.6
	2	3.85	0.80	4.06	47.1
	3	11.4	0	3.50	0
	4	3.66	2.42	2.14	15.1
	5	12.10	0	5.68	0

用费希尔判别法判别样本 $x_0=[7.90 \ 2.40 \ 4.30 \ 33.2]$ 和 $x_1=[12.40 \ 5.10 \ 4.48 \ 24.6]$ 分别属于哪一类。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
XA=[13.5 2.79 7.8 49.6;22.31 4.67 12.31 47.8;28.82 4.63 16.18 62.15;...
15.29 3.45 7.58 43.2;28.29 4.90 16.12 58.7];
XB=[2.18 1.06 1.22 20.6;3.85 0.80 4.06 47.1;11.4 0 3.50 0;...
3.66 2.42 2.14 15.1;12.10 0 5.68 0];
x0=[7.90 2.40 4.30 33.2];
x1=[12.40 5.10 4.48 24.6];
k=Fisherbanbie(XA,XB,x0)
样品属于第二类
k =
2
>> k=Fisherbanbie(XA,XB,x1)
样品属于第一类
k =
1
```

所以 x_0 属于第二类，而 x_1 属于第一类。

9.3 试验分析

9.3.1 试验相关概述

化学实验设计和优化是数理统计方法在化学中应用比较成熟的一个领域。

实验设计是指在实验各影响因素的取值范围内，最有效地选择实验点，科学地安排实验，进而通过数据分析得到指标取得最优值条件的一种方法，即研究如何设计实验条件使指标获得最优值。一个好的实验设计应能以最小的实验工作回答所有有关研究对象的问题。MATLAB 中介绍了完全析因设计、不完全析因设计和 D-优化设计三种实验设计方法。

1. 完全析因设计

为了在几个水平上研究几个因素而设计的实验称为析因实验设计，它不仅要研究各因素水

平指标的影响,而且还强调分析诸因素对指标的作用。它是按析因设计表设计方案,通过分析实验指标的变化决定各因素主效应和各因素之间的实验方法。

在 MATLAB 中提供了一些函数实现完全析因设计,介绍如下。

1) ff2n 函数

功能: 二水平完全析因分析。其调用格式如下:

dFF2 = ff2n(n): 创建一个二水平的完全析因设计 X。

【例 9-15】二水平完全析因分析示例。

```
>> dFF2 = ff2n(3)
dFF2 =
    0    0    0
    0    0    1
    0    1    0
    0    1    1
    1    0    0
    1    0    1
    1    1    0
    1    1    1
```

2) fullfact 函数

功能: 完全析因试验设计。其调用格式如下:

dFF = fullfact(levels): 给定因子设置,进行完全析因设计。**levels** 向量中的每一个元素指定 **design** 对应列中唯一元素的个数。

【例 9-16】完全析因试验设计示例。

```
>> dFF = fullfact([3 5])
dFF =
    1    1
    2    1
    3    1
    1    2
    2    2
    3    2
    1    3
    2    3
    3    3
    1    4
    2    4
    3    4
    1    5
    2    5
    3    5
```

2. 不完全析因分析

完全析因设计的困难之一是当变量增加时,进行析因设计的组合将呈指数增长(2^n)。因此完全析因分析一般只适合于因素和水平较少的实验。当有较多因素及其水平的析因分析时,可以采用不完全析因试验设计。

不完全析因分析可以通过较少的试验研究每个变量的主效应,可以大大减少实验次数。例

如当变量为 7 时，完全析因实验次数将达到 128 次，而不完全析因分析则只需 8 次。

在 MATLAB 中提供了 `fracfact` 函数实现不完全析因设计。

功能：生成源于生成器的不完全析因分析。其调用格式如下：

`dfF = fracfact(generators)`：根据生成器字符串 `generators` 指定的内容生成不完全析因设计，并返回设计点的矩阵 `dfF`。

`[dfF,confounding] = fracfact(generators)`：返回一个单元混淆阵列 `confounding`。

【例 9-17】不完全析因分析示例。

```
>> generators = fracfactgen('a b c d',3,4)
generators =

    'a'
    'b'
    'c'
    'abc'

>> [dfF,confounding] = fracfact(generators)
dfF =

    -1    -1    -1    -1
    -1    -1     1     1
    -1     1    -1     1
    -1     1     1    -1
     1    -1    -1     1
     1    -1     1    -1
     1     1    -1    -1
     1     1     1     1

confounding =

    'Term'    'Generator'    'Confounding'
    'X1'      'a'          'X1'
    'X2'      'b'          'X2'
    'X3'      'c'          'X3'
    'X4'      'abc'        'X4'
    'X1*X2'   'ab'         'X1*X2 + X3*X4'
    'X1*X3'   'ac'         'X1*X3 + X2*X4'
    'X1*X4'   'bc'         'X1*X4 + X2*X3'
    'X2*X3'   'bc'         'X1*X4 + X2*X3'
    'X2*X4'   'ac'         'X1*X3 + X2*X4'
    'X3*X4'   'ab'         'X1*X2 + X3*X4'
```

3. D-优化设计

不完全析因设计和熟知的正交实验，由于具有“均匀分散、整齐可比”的特点，可以用较少的实验获得各因素及其相互之间作用的丰富信息。但是为了达到“整齐可比”的目的，往往要做较多的实验（至少为水平数的平方）。若各因素取 5 个水平，则至少要做 $5^2 = 25$ 次试验，这在实际应用中较难实现。

为此，必须寻找一种适用于多因素多水平而实验次数更少的实验设计方案，20 世纪 70 年代出现的 D-优化设计便是其中的一种。D-优化设计使费希尔信息矩阵 $\mathbf{X}^T \mathbf{X}$ 的行列式最大化，该矩阵与参数的协方差矩阵的逆成比例，所以 $\det(\mathbf{X}^T \mathbf{X})$ 等价于使参数协方差矩阵的行列式最大化。

在 MATLAB 中提供了一些函数实现 D-优化设计, 分别介绍如下。

1) cordexch 函数

功能: 协同交换算法。其调用格式如下:

`dCE = cordexch(nfactors,nruns)`: 生成因子设置矩阵 `dCE`, `nruns` 为实验次数。

`[dCE,X] = cordexch(nfactors,nruns)`: 生成相关的设计矩阵 `X`。

`[dCE,X] = cordexch(nfactors,nruns,model)`: 指定的回归模型进行设计, 输入参数 `model` 可以是 'interaction'、'quadratic' 或 'purequadratic'。

`[dCE,X] = cordexch(...,param1,val1,param2,val2,...)`: 设置回归模型设计的参数及其参数值。

【例 9-18】协同交换算法示例。

```
>> nfactors = 3;
nruns = 7;
[dCE,X] = cordexch(nfactors,nruns,'interaction','tries',10)
dCE =
    -1    -1     1
     1     1     1
     1    -1     1
    -1     1    -1
    -1    -1    -1
     1     1    -1
    -1     1     1
X =
     1    -1    -1     1     1    -1    -1
     1     1     1     1     1     1     1
     1     1    -1     1    -1     1    -1
     1    -1     1    -1    -1     1    -1
     1    -1    -1    -1     1     1     1
     1     1     1    -1     1    -1    -1
     1    -1     1     1    -1    -1     1
```

2) daugment 函数

功能: 试验设计的 D-优化扩展。其调用格式如下:

`dCE2 = daugment(dCE,mruns)`: 扩展一个初始实验设计 `dCE`, 并进行 `n` 次新的测试。

`[dCE2,X] = daugment(dCE,mruns)`: 生成相关的设计矩阵 `X`。

`[dCE2,X] = daugment(dCE,mruns,model)`: 输入 `model` 控制回归模型的阶次。

【例 9-19】试验设计的 D-优化扩展示例。

```
>> dCEmain = cordexch(4,8)
dCEmain =
     1    -1     1     1
     1     1     1    -1
    -1     1    -1     1
     1     1    -1    -1
    -1    -1     1    -1
     1    -1    -1    -1
    -1    -1    -1    -1
     1    -1    -1     1
>> dCEinteraction = daugment(dCEmain,8,'interaction')
```

dCEinteraction =

```

1    -1    1    1
1     1    1   -1
-1     1   -1    1
1     1   -1   -1
-1   -1    1   -1
1    -1   -1   -1
-1   -1   -1   -1
1    -1   -1    1
-1   -1   -1    1
1     1    1    1
1    -1    1   -1
-1     1    1    1
-1     1    1   -1
-1   -1    1    1
-1     1   -1   -1
1     1   -1    1

```

3) dcovary 函数

功能：用指定的协方差进行 D-优化设计。其调用格式如下：

dCV = dcovary(nfactors, fixed): 为每一次运行创建一个有固定协变量约束的 D-优化设计。

[dCV, X] = dcovary(nfactors, fixed): 生成相关的设计矩阵 X。

[dCV, X] = dcovary(nfactors, fixed, model): model 控制回归模型的阶次。默认时为一线性模型。

[dCV, X] = daugment(..., param1, val1, param2, val2, ...): 设置回归模型设计的参数及其参数值。

【例 9-20】用指定的协方差进行 D-优化设计示例。

```
>> fixed = dummyvar([1 1 2 2 3 3 4 4]);
```

```
dCV2 = dcovary(2, fixed(:, 1:3), 'linear')
```

dCV2 =

```

-1    -1     1     0     0
1     1     1     0     0
1    -1     0     1     0
-1     1     0     1     0
-1   -1     0     0     1
1     1     0     0     1
-1     1     0     0     0
1    -1     0     0     0

```

4) hadamard 函数

功能：Hadamard 矩阵。其调用格式如下：

H = hadamard(n): 返回阶次为 n 的 Hadamard 矩阵。

【例 9-21】生成 Hadamard 矩阵。

```
>> hadamard(4)
```

ans =

```

1     1     1     1
1    -1     1    -1
1     1    -1    -1
1    -1    -1     1

```


5) rowexch 函数

功能：试验设计的 D-优化设计—行交换算法。其调用格式如下：

$\text{dRE} = \text{rowexch}(\text{nfactors}, \text{nruns})$ ：生成因子设置矩阵 dRE ，用带常数项的线性累加模型进行 D-优化设计。

$[\text{dRE}, \text{X}] = \text{rowexch}(\text{nfactors}, \text{nruns})$ ：生成相关的设计矩阵 X 。

$[\text{dRE}, \text{X}] = \text{rowexch}(\text{nfactors}, \text{nruns}, \text{model})$ ：为拟合指定的回归模型生成设计。输入 model 为控制回归模型的阶次。

$[\text{dRE}, \text{X}] = \text{rowexch}(\dots, \text{param1}, \text{val1}, \text{param2}, \text{val2}, \dots)$ ：设置回归模型设计的参数及其参数值。

【例 9-22】试验设计的 D-优化设计—行交换算法示例。

```
>> nfactors = 3;
nruns = 7;
[dRE,X] = rowexch(nfactors,nruns,'interaction','tries',10)
dRE =
    -1     1    -1
     1     1     1
    -1    -1     1
    -1    -1    -1
     1    -1     1
     1     1    -1
     1    -1    -1
X =
     1    -1     1    -1    -1     1    -1
     1     1     1     1     1     1     1
     1    -1    -1     1     1    -1    -1
     1    -1    -1    -1     1     1     1
     1     1    -1     1    -1     1    -1
     1     1     1    -1     1    -1    -1
     1     1    -1    -1    -1    -1     1
```

9.3.2 试验分析的实现

上面已经介绍了各函数的调用格式及其基本用法，下面将举例子说明各函数在实际中的应用。

【例 9-23】考察反应： $A \xrightarrow[H^+]{\Delta} B + C$ ，研究 A 的浓度及反应温度对产率的影响。请以 D-优化设计方法分析各因素的主效应和交互效应。

考虑二输入交互模式，使用行交换算法进行实验设计，该模型的形式为

$$y = a_0 + a_1x_1 + a_2x_1x_2 + e(\text{误差})$$

假设希望该 D-优化设计通过四次实验来拟合模型，其实现的 MATLAB 程序代码如下：

```
>> [dRE,X] = rowexch(2,4,'i')
dRE =
     1    -1
     1     1
    -1    -1
    -1     1
```

X =

```

1      1      -1     -1
1      1       1      1
1     -1     -1      1
1     -1      1     -1
    
```

据此，设计实验点并得到表 9-4 所列的实验结果。

表 9-4 二因素二水平 D-优化设计表

实验序号	I	A	B	AB	指标 Y
1	+1	+1	-1	-1	80.4
2	+1	-1	-1	+1	72.4
3	+1	+1	+1	+1	94.4
4	+1	-1	+1	-1	90.6

表中第二列是为了分析各个因素对指标的影响，都以高水平表示；第三、四列为 A 、 B 两因素的实验点，第四列为它们之间的交互作用。

根据 D-优化设计 X 矩阵和实验结果矩阵 Y ，可得到系数矩阵

$$A = x^{-1} \cdot Y = n^{-1} \cdot X^T \cdot Y$$

对于本题，有

```

>> x=[+1 +1 -1 -1;+1 -1 -1 +1;+1 +1 +1 +1;+1 -1 +1 -1];
y=[80.4 72.4 94.4 90.6];
A=1/4*x'*y'
A =
    84.4500
     2.9500
     8.0500
    -1.0500
    
```

即 A 的主效应为 2.9500，是正效应但值不大，对指标的影响不大； B 的主效应为 8.050 0，影响最大；交互作用为-1.0500，影响较小，可以认为基本上不存在交互作用。当然也可以进行方差分析以求出各影响因素。

【例 9-24】影响分光光度法测定的因素有 pH 值、反应物和显色剂及其他掩蔽剂（或强度调节剂等）浓度、反应温度等因素，请设计一个实验方案，以最少的实验次数达到最佳的结果。

这一个多因素、多水平的实验，不适合采用完全析因实验设计方法。为了尽量减少实验次数，现采用 D-优化设计。

假设本例中有 m 个因素，至少做 n 次试验。先不考虑三次项和三因素之间的交互作用，此时，指标函数的回归方程形式为

$$y = a_0 + \sum_{i=1}^m a_i x_i + \sum_{i=1}^m \sum_{j=1}^m x_i x_j$$

以矩阵形式表示即为

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1T} \\ 1 & x_{21} & \cdots & x_{2T} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n2} & \cdots & x_{nT} \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_T \end{bmatrix}$$

其中, $T = m + 0.5m(m+1)$ 。

应用 $m=4$ (pH、 T 、反应物和显色剂浓度), 所以 $T=14$, 则为了满足最小二乘的条件, 至少需做 15 次实验左右。此时, 利用 rowexch 函数便可进行 D-优化设计。

调用实验次数, 可得到不同的实验设置。根据 rowexch 函数计算结果, 可得出: $11 \leq n \leq 15$ 。即在考虑交互作用时, 至少需做 11 次实验, 各实验的输入如下:

```
>> [dRE,X] = rowexch(4,11,'t');
>> dRE
dRE =
    -1    -1     1    -1
     1     1     1    -1
    -1    -1     1     1
     1    -1    -1     1
     1     1    -1    -1
    -1    -1    -1    -1
     1     1     1     1
     1    -1    -1    -1
    -1     1     1    -1
    -1     1    -1     1
     1    -1     1     1
```

X 矩阵则是为拟合上述回归方程的设计矩阵。根据 dRE 矩阵安排实验点, 得到指标矩阵 Y , 则可以根据多元回归模型 (或方差分析) 求出回归方程, 从而找出主要影响因素和最佳实验点。

如果不考虑交互作用, 则回归模型为

$$y = a_0 + \sum_{i=1}^m a_i x_i + \sum_{j=1}^m x_j^2$$

则 $T = m + m = 8$, 此时只需做 9 次左右的实验, 同样利用 rowexch 或 cordexch 函数进行优化设计:

```
>> [dRE,X] = rowexch(4,11,'p');
>> dRE
dRE =
    -1    -1    -1     0
     0     1    -1     1
    -1     0     1    -1
     1     0    -1    -1
     1    -1     0     1
    -1     1     0     1
     1     1     1     0
     0    -1     0    -1
     0    -1     1     1
```

0	0	0	0
0	1	0	-1

调整实验次数，比如 `cordexch` 函数的计算结果，可知此时最少应做 9 次实验，各次实验安排如 *dRE* 矩阵。

9.4 正交实验设计

正交试验设计是多因素的优化试验设计方法。它从全面实验的样本点中挑选出部分有代表性的样本点做试验，这些代表点具有正交性。其作用是指用较少的试验次数就可以找出因素水平之间的最优搭配或由试验结果通过计算推断出最优搭配。

用正交表安排实验首先看因素的水平，选取与因素水平相同的正交表，然后看因素的数目，因素的个数不能超过正交表的列数，允许有空白列（即不安排实验）。根据正交表安排好实验后，采用抽签的方法随机选取实验顺序，并且在实验中要尽量保持实验因素以外的其它因素固定，在不能避免的场合可以增加一个区组因素，也安排在正交表的一个列上，如人员的差异。此时可以把人看成一个因素。

正交实验后得到的数据可以采用两种方法处理，一种是直观极差分析（直观分析法），另外一种 是方差分析法。

9.4.1 极差分析

下面通过一个例子来说明极差分析法。

【例 9-25】某化工厂生产一种产品，产率较低。现在希望通过实验设计，找出好的生产方案，以提高产率。影响产率的因素见表 9-5。

表 9-5 因素与水平

水平	因素		
	A (反应温度/℃)	B (加碱量/kg)	C (催化剂种类)
1	79	36	甲
2	86	45	乙
3	91	54	丙

根据影响因素及每个因素的水平数，选择 L_9 （ L_9 为实验的因素数）正交表安排实验，并得到表 9-6 所列的实验结果。对表中的数据进行分析可得到 T 、 \bar{T} 和 R ，其中 T 为各因素同一水平结果之和， \bar{T} 为其平均值， R 为极差。

表 9-6 实验结果

实验号	因素				实验结果
	A	B	C	空白列	
1	1(80)	1(35)	1(甲)	1	51
2	1	2(48)	2(乙)	2	72
3	1	3(55)	3(丙)	3	58
4	2(85)	1	2	3	82

续表

实验号	因素				实验结果
	A	B	C	空白列	
5	2	2	3	1	69
6	2	3	1	3	59
7	3(90)	1	3	2	77
8	3	2	1	3	85
9	3	3	2	1	84

1) 直接编程实现极差分析

其实现的 MATLAB 程序代码如下:

```
>> clear all;
data1=[1 1 1 51;1 2 2 71;1 3 3 58;2 1 2 82;2 2 3 69;2 3 1 59;...
3 1 3 77;3 2 1 85;3 2 2 84];
f=3;r=3;
[r1,c]=size(data1); t=zeros(f,r);
for k=1:f
    for j=1:r
        b=0;
        for i=1:r1
            if data1(i,j)==k %水平相同
                b=b+data1(i,c);
            end
        end
        t(k,j)=b;
    end
end
t1=t/3, t
r=max(t1)-min(t1),
```

运行程序, 输出如下:

```
t1 =
    60    70    65
    70   103    79
    82    39    68

t =
   180   210   195
   210   309   237
   246   117   204

r =
    22    64    14
```

从结果中可看出, 理论上最优方案为 $B_2A_3C_2$, 最大的影响因素为 A, 即温度。

2) 编写函数实现极差分析

以上的程序代码是针对分析数据直接编写出的代码, 下面还可以通过编写极差分析函数来实现。把编写的极差分析函数保存在 MATLAB 目录下, 以后用到极差分析直接调用此函数。

无交互作用的极差分析 jcfxhshu.m 的源代码如下:

```
function y=jcfxhshu(A)
[m,n]=size(A);
B=A(:,1:end-1);
mm=max(B(:));
K=zeros(mm,n-1);
for kh=1:m
    for k1=1:(n-1)
        kt=A(kh,k1);
        K(kt,k1)=K(kt,k1)+A(kh,end);
    end
end
K(K==0)=NaN;
y.K=K;
M=max(B);
MM=m./M;
Km=K./MM(ones(mm,1),:);
y.m=Km;
[tem,you]=max(Km);
y.you=you;
R=max(Km)-min(Km);
y.R=R;
[temp,Cx]=sort(R);
y.cixu=fliplr(Cx);
```

这一函数的调用格式是 $y = jcfxhshu(A)$ 。输入参数 A 是一个数据矩阵。 A 的最后一列是试验结果，其余部分是正交表。输出参数 y 是一个结构数组：第一个域 K 是 K 值；第二个域 m 是 K 值的平均值；第二个域 you 是优水平；第三个域 R 是极差 R ；第四个域是 $cixu$ 是主次顺序。

下面对例 9-25 的数据进行极差分析。其程序代码如下：

```
>> y=jcfxhshu(data1)
```

运行程序，输出如下：

```
y =
    K: [3x3 double]
    m: [3x3 double]
  you: [3 2 2]
    R: [22 64 14]
  cixu: [2 1 3]
```

从运行的结果可以看出，因素的主次顺序是 $2 \rightarrow 1 \rightarrow 3$ 。即为 $B \rightarrow A \rightarrow C$ 。再从优水平知，因素 A 的优水平是 3；因素 B 的优水平是 2；因素 C 的优水平是 2，所以最优组合是 $B_2A_3C_2$ 。

3) 趋势图绘制函数实现极差分析

编写的趋势图绘制函数为 $qstzhzshu.m$ 。其源代码如下：

```
function qstzhzshu(K)
[m,n]=size(K);
nk=ceil(sqrt(n));
mk=ceil(n/nk);
figure;
for kk=1:n
    subplot(mk,nk,kk)
```

```

plot(K(:,kk),'r-');
SS=['因素',num2str(kk)];
title(SS);
end

```

这一函数的输入参数是由极差分析函数得到的 K 值，运行后得到各因素的趋势图。也可以用极差分析函数得到的 m 值，作为函数的输入参数，绘制趋势图。

对例 9-25，由上面的极差分析得到的分析结果 y ，继续在命令窗口输入：

```
>> qstzhshu(y.K)
```

运行程序，得到的趋势图如图 9-8 所示。

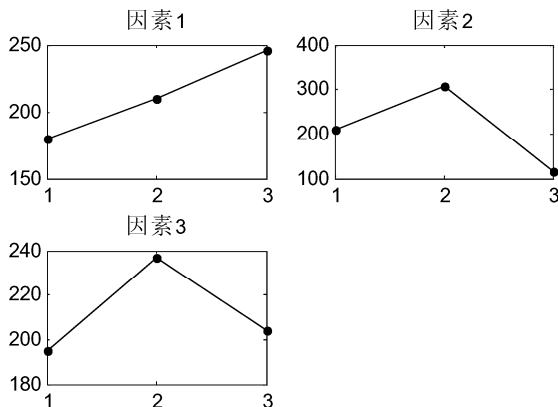


图 9-8 趋势图

9.4.2 方差分析

直观分析虽然比较简便易懂，但不能估计试验误差的大小，很难断定因素的重要性。为了克服这个缺点，可采用方差分析的方法。

1. 编程实现方差分析

下面对例 9-25 的数据进行方差分析。其程序代码如下：

```

>> g={ [1 1 1 2 2 2 3 3 3]; [1 2 3 1 2 3 1 2 3]; [1 2 3 2 3 1 3 1 2] };
anovan(data1(:,c),g)' %多因素方差分析

```

运行程序，输出如下，其方差分析表如图 9-9 所示。

```

ans =
    0.1057    0.4674    0.2087

```

Figure 9-9 shows the ANOVA table output from the software. The table is titled "Analysis of Variance" and includes the following data:

Source	Sum Sq	d.f.	Mean Sq	F	Prob>F
X1	728	2	364	8.47	0.1057
X2	98	2	49	1.14	0.4674
X3	326	2	163	3.79	0.2087
Error	86	2	43		
Total	1238	8			

Constrained (Type III) sums of squares.

图 9-9 方差分析表

因为三个因素的 P 值都大于 0.05, 不能断定 3 个因素都不显著, 而是要剔除一个最不显著的因素。在此例中剔除 B , 然后再做方差分析。

```
>> g1={1 1 1 2 2 2 3 3 3};[1 2 3 2 3 1 3 1 2]};
anovan(data1(:,4)',g1)' %多因素方差分析
```

运行程序, 输出如下, 输出的方差分析表如图 9-10 所示。

```
ans =
    0.0407    0.1302
```

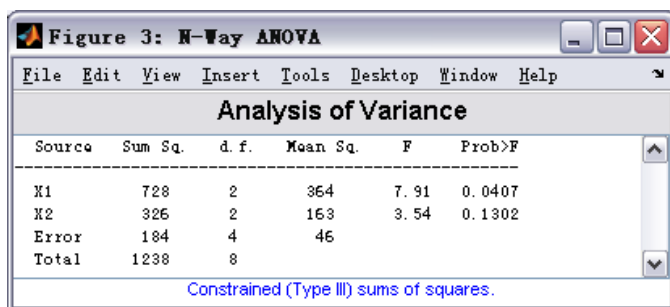


图 9-10 剔除 B 后的方差分析表

从上面的分析可确定 A 是重要因素, C 是次要因素。

2. 自定义编写方差分析函数 fcfxi.m

其实现的 MATLAB 程序代码如下:

```
function table=fcfxi(A,z0)
x=A(:,end);
A(:,end)=[];
[n,k]=size(A);
bz=1:k;
if nargin==2
    z0=z0(:);
    if (~isnumeric(z0))|(any(floor(z0)~=z0))|(max(z0)>k)|(any(z0<1))
        error('空列标志错误');
    end
end
mm=max(A(:));
K=zeros(mm,k);
for kh=1:n
    for kl=1:k
        kt=A(kh,kl);
        K(kt,kl)=K(kt,kl)+x(kh);
    end
end
alpha1=0.05; alpha2=0.01;
m=max(A);
r=n./m;
SST=(x-mean(x))'*(x-mean(x));
Km=K./r(ones(mm,1),:);
Kmm=(Km-mean(x)).*(Km-mean(x));
Kmm(K==0)=0;
```



```

SSj=sum(Kmm,1).^r;
fT=n-1;
fy=(mm-1)*ones(k,1);
tsf=[(1:k)',SSj',fy,(1:k)'];
if nargin==2
    tsf(end+1,:)=sum(tsf(z0,:),1);
    tsf(z0,:)=[];
    Ve=tsf(end,2)/tsf(end,3);
end
if nargin==1
    [tem,z0]=min(SSj);
    Ve=SSj(z0)/fy(z0);
end
V=tsf(:,2)./tsf(:,3);
Se=0; fe=0;
for kk=1:length(V)
    if(V(kk)<=2*Ve)
        Se=Se+tsf(kk,2);
        fe=fe+tsf(kk,2);
        tsf(kk,4)=0;
    end
end
Ve=Se/fe;
Fb=V/Ve;
[m1,tem]=size(tsf);
table=cell(m1+3,7);
table(1,:)={'方差来源','平方和','自由度','均方差','F 值','Fa','显著性'};
for kkk=1:m1
    if tsf(kkk,4)==0
        table{kkk+1,1}=['因素',num2str(tsf(kkk,1)),'*'];
    else
        table{kkk+1,1}=['因素',num2str(tsf(kkk,1))];
    end
end
if (tsf(m1,4)==0)&(nargin==2)
    table{m1+1,1}=['误差*'];
end
M=[tsf(:,[2,3]),V,Fb];
for kh=2:(m1+1)
    for kl=2:5
        table{kh,kl}=M(kh-1,kl-1);
    end
end
ntst=length(Fb);
for ktst=1:ntst
    F=finv(1-[alpha1;alpha2],tsf(ktst,3),fe);
    F1=min(F);
    F2=max(F);

```

```

table{ktst+1,6}=[num2str(F1),',';num2str(F2)];
if Fb(ktst)>F2
    table{ktst+1,7}='高度显著';
elseif (Fb(ktst)<=F2)&(Fb(ktst)>F1)
    table{ktst+1,7}='显著';
end
end
table(end-1,1:4)={'误差',Se,fe,Ve};
if table{end-2,3}==fe
    table(end-2,:)=[];
end
table(end,1:3)={'总和',SST,n-1};
    
```

这一函数有两个输入参数，第一个输入参数 **A** 是数据矩阵，其最后一列是试验结果，去掉 **A** 最后一列剩余部分就是试验所用的正交表。第二个输入参数 **z0** 表示空列的列号，是一个向量，如果没有空列，可以不输入，这里程序会自动把平方和最小的一个作为误差。

下面对例 9-25 的数据进行方差分析。其程序代码如下：

```

>> data1=[1 1 1 51;1 2 2 2 71;1 3 3 3 58;2 1 2 3 82;2 2 3 1 69;2 3 1 3 59;...
    3 1 3 2 77;3 2 1 3 85;3 2 2 1 84]; %把空列补上
table=fcfxi(data1,4)
    
```

运行程序，输出如下：

```

table =
    '方差来源'    '平方和'    '自由度'    '均方差'
    '因素 1*'    [1.4290e+007]    [    2]    [7.1450e+006]
    '因素 2'    [8.5983e+009]    [    2]    [4.2992e+009]
    '因素 3*'    [1.2832e+006]    [    2]    [6.4159e+005]
    '误差*'    [1.1191e+009]    [    2]    [5.5955e+008]
    '误差'    [1.1347e+009]    [1.1347e+009]    [    1]
    '总和'    [    1238]    [    8]    []

    'F 值'    'Fa'    '显著性'
    [7.1450e+006]    '2.9684;4.6102'    '高度显著'
    [4.2992e+009]    '2.9684;4.6102'    '高度显著'
    [6.4159e+005]    '2.9684;4.6102'    '高度显著'
    [5.5955e+008]    '2.9684;4.6102'    '高度显著'
    [    ]    [    ]    [    ]
    [    ]    [    ]    [    ]
    
```

从运行结果可以看出，三个因素对试验的结果影响是“高度显著”，也就是对试验的结果都有影响；由于其均方差的值没有超过空列误差的 2 倍，所以把它的误差加到空列误差中，得到新的误差。

第 10 章 工程数据分析中的应用

应用最优技术解决实际工程问题称为工程优化技术。最优化方法是运筹学的一个重要组成部分，在自然科学、社会科学、生产实践、工程设计中有着重要的实用价值和参考意义。利用最优化技术，可以实现生产成本最小化、运行效率最大化、车辆路线最短化等。

10.1 工程优化问题的概述

在现代工程设计、经济管理和市场规划等领域，广泛涉及工程优化问题。对于工厂企业，如何在消耗总工时最小的情况下获取最大的产品数量？如何安排物流秩序，在满足最大效率的前提下，达到成本最低、运费最少？工程优化问题几乎涉及社会生活的每一个领域。对于工程优化问题，利用最优化理论和方法进行求解，帮助决策者作出最优的决策，以最低的成本，获取最高的利润。

假设某种物资有 m 个产地， n 个销地。第 i 个产地的产量为 $a_i (i=1, 2, \dots, m)$ ，第 j 个销地的需求量为 $b_j (j=1, 2, \dots, n)$ 。考虑到实际情况，产量不小于销量，即满足 $\sum_{i=1}^m a_i \geq \sum_{j=1}^n b_j$ ，由产地 i 前往销地 j 的运输成本为 p_{ij} ，那么如何安排运输才能既满足各地的需求，同时使花费的运输成本最低？

对于这样一个工程优化问题，给出了实际问题的背景，利用最优化理论和方法来求解实际生产问题时，都需要将实际问题进行抽象化，并且加以简化，提取问题的核心，建立数学模型，此时需要决策者确定问题的决策变量、决策目标以及各个不同的约束条件，形成最优化数学模型；虽然对实际问题进行了抽象简化，但有些时候建立的模型比较复杂，不利于数学手段的求解，还需要对数学模型进行变换，使之更加简洁，并且能够被求解；最后还要分析求解结果是否符合实际情况。下面将以这样的步骤分析运输问题。

1) 选取决策变量

以产地 i 运往销地 j 的货物数量 x_{ij} 为决策变量。

2) 确定决策目标

运输问题要求运输成本最低，问题的决策目标为运输的总成本 $z = \sum_{i=1}^m \sum_{j=1}^n p_{ij} x_{ij}$ 。

3) 确定约束条件

在约束条件中，首先必须保证运输货物能够满足各地的需求，即产地运往销地 j 的总量为 b_j ，得到各销地需求量的等式约束： $\sum_{i=1}^m x_{ij} \leq b_j (j=1, 2, \dots, n)$ 。同时从第 i 个产地运出的货物总

数不能超过其产量，得到的产量不等式约束： $\sum_{j=1}^n x_{ij} \leq a_i (i=1, 2, \dots, m)$ 。另外，决策变量自身取值范围的约束，即 $x_{ij} \geq 0 (i=1, 2, \dots, m; j=1, 2, \dots, n)$ 。于是运输问题就可以转化为以下数学模型的最优化问题：

$$\begin{aligned} \min z &= \sum_{i=1}^m \sum_{j=1}^n p_{ij} x_{ij} \\ s.t. &\begin{cases} \sum_{i=1}^m x_{ij} = b_j (j=1, 2, \dots, n) \\ \sum_{j=1}^n x_{ij} \leq a_i (i=1, 2, \dots, m) \\ x_{ij} \geq 0 (i=1, 2, \dots, m; j=1, 2, \dots, n) \end{cases} \end{aligned}$$

对该数学模型进行求解，就可以得到决策变量：从产地 i 运往销地 j 的货物数量 x_{ij} 。此时，一方面能够满足各地的需求，另一方面使总的运输成本最低。

通过上面简单的一个运输问题可以看出，工程优化问题存在于实际生活的每一个角落，因此掌握工程优化问题数学建模方法和求解，对于决策者将起到至关重要的作用。在实际问题中，不同问题所设计的数学模型不同，没有统一的方法进行数学建模，但是对于建立好的数学模型，只有经过求解，得到求解结果后，应用于实际生活中，才能体现出求解结果的优劣程度。

10.2 线性优化问题

10.2.1 线性优化问题的基本知识

规划问题研究的对象大体可分为两大类：一类是在现有的人、财、物等资源的条件下，研究如何合理地计划、安排，使得某一目标达到最大，如产量、利润目标等；另一类是在任务确定后，如何计划、安排，使用最低限度的人、财等资源，去实现该任务，如使生产成本、费用最低等。这两类问题从本质上是相同的，即都在一组约束条件下，实现某一目标的最优（最大或最小）。线性规划研究的问题要求目标与约束条件函数均是线性的，而目标函数只能是一个。在经济管理问题中，大量的问题是线性的，其他大多也可以转化为线性，从而使线性规划有极大的应用价值。线性规划模型包括 3 个要素：

- (1) 决策变量。问题中需要求解的那些未知量，一般用 n 维向量 $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ 表示。
- (2) 目标函数。通常是问题中需要优化的那个目标的数学表达式，它是决策变量 \mathbf{x} 的线性函数。
- (3) 约束条件。对决策变量的限制条件，即 \mathbf{x} 的允许取值范围，它通常是 \mathbf{x} 的一组线性不等式或线性等式。

线性规划问题的数学模型一般可表示如下：

目标函数

$$\min(\max)z = \sum_{j=1}^n c_j x_j \quad (10-1)$$

约束条件

$$\begin{cases} \sum_{j=1}^n a_{ij} x_j \leq b_i & (i=1,2,\dots,m) \\ x_j \geq 0 & (j=1,2,\dots,n) \end{cases} \quad (10-2)$$

这里 z 称为目标变量, a_{ij} 、 c_j 、 b_i ($i=1,2,\dots,m; j=1,2,\dots,n$) 是常数, 式 (10-1) 称为目标函数, 式 (10-2) 称为约束条件, 简记为 $s.t.$ 。

称满足约束条件 (10-2) 的解 $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ 为线性规划问题的可行解, 而使目标函数式 (10-1) 达到最小值的可行解叫最优解。

运用矩阵知识, 式 (10-1) 与式 (10-2) 可用矩阵与向量的形式表示为

$$\min z = \mathbf{c}^T \mathbf{x} \quad (10-3)$$

$$s.t. \begin{cases} \mathbf{Ax} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{cases} \quad (10-4)$$

式中: $\mathbf{A} = (a_{ij})_{m \times n}$ 称为技术系数矩阵; $\mathbf{b} = (b_1, b_2, \dots, b_m)^T$ 称为资源系数向量; $\mathbf{c} = (c_1, c_2, \dots, c_n)^T$ 称为价值系数向量; $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ 称为决策向量。

在 MATLAB 优化工具箱提供了许多优化函数迭代过程显示、最大迭代次数设置等一系列的选型设置。通过 `optimset` 和 `optimget` 函数即可获得当前优化工具箱的默认属性设置。在进入优化函数介绍前, 对这些属性设置进行简单的介绍。

在命令窗口中输入:

```
>> optimset %获取 MATLAB 优化工具箱所有的属性设置选项
      Display: [ off | iter | iter-detailed | notify | notify-detailed | final | final-detailed ]
      MaxFunEvals: [ positive scalar ]
      MaxIter: [ positive scalar ]
      TolFun: [ positive scalar ]
      TolX: [ positive scalar ]
.....
TolProjCGAbs: [ positive scalar | { 1e-10} ]
      TolRLPFun: [ positive scalar | { 1e-6} ]
      TolXInteger: [ positive scalar | { 1e-8} ]
      TypicalX: [ vector | {ones(numberOfVariables,1)} ]
      UseParallel: [ always | {never} ]
```

或者在命令窗口中输入:

```
>> optimset fmincon %获取 fmincon 优化函数当前的默认属性值
ans =
      Display: 'final'
      MaxFunEvals: []
      MaxIter: []
.....
TolProjCGAbs: 1.0000e-010
```

```
TolRLPFun: []
TolXInteger: []
TypicalX: 'ones(numberofvariables,1)'
UseParallel: 'never'
```

通过 `optimset` 函数可以获取优化函数的属性参数。

1. optimset 函数

功能：用来获取或者设置优化函数的属性选项结构。其调用格式如下：

`options = optimset('param1',value1,'param2',value2,...)`：创建一个包含设定属性选项的空的 `optimset` 属性结构体。

`optimset`：没有任何输入参数和返回参数，得到优化工具箱属性选项结构体及属性值的可行值和取值类型。

`options = optimset`：返回属性选项结构体，其中属性值全为空集。

`options = optimset(optimfun)`：获取具有优化函数的默认属性选项结构体。

`options = optimset(oldopts,'param1',value1,...)`：复制 `oldopts` 的所有属性选项，并且修改设定属性选型。

`options = optimset(oldopts,newopts)`：将 `newopts` 所有属性值覆盖 `oldopts` 对应的属性选项的属性值。

【例 10-1】如果想修改 `fmincon` 函数中属性选项 `Display` 为 `Iter`，`TolFun` 属性选项为 `1e-8`。其实现的程序代码如下：

```
>> oldopts = optimset('fmincon'); %获取 fmincon 函数的属性选项结构体
>> newopts=optimset(oldopts,'Display','iter','TolFun',1e-8); %设置属性结构体,并保存为 newopts
>> optimget(newopts,'Display') %获取新结构体中 Display 的属性值
ans =
    iter
```

同样可以使用以下方法：

```
>> options=optimset('Display','iter','TolFun',1e-8);
```

2. optimget 函数

功能：`optimget` 函数和 `optimset` 函数相对应，用来获取属性结构体中属性选项的属性值。其调用格式如下：

`val = optimget(options,'param')`：获取 `options` 中给定属性选型的属性值。

`val = optimget(options,'param',default)`：获取 `options` 属性中给定属性选项的默认属性值。

【例 10-2】`optimget` 函数用法。

```
>> options=optimset('fminbnd'); %获取 fminbnd 函数的属性结构体
>> val=optimget(options,'Display','iter') %获取 fminbnd 函数默认的 Display 属性值
val =
    notify
>> options=optimset(options,'Display','iter'); %修改 Display 属性值为 iter
>> val=optimget(options,'Display') %重新获取 fminbnd 函数默认的 Display 属性值
val =
    iter
```

`options` 属性结构体中共有 51 个属性选项，其中有些可适用于大中规模优化问题，有些仅适用于大规模求解问题或者中规模求解问题。如果读者对这些属性选项感兴趣，可以查阅相关的帮助文档。

10.2.2 线性规划的 MATLAB 实现

在 MATLAB 优化工具箱中,使用 `linprog` 函数对线性优化问题进行求解,其调用格式如下:

`x = linprog(f,A,b)`: 求 $\min c'*x$ 在约束条件 $A*x \leq b$ 下线性规划的最优解。

`x = linprog(f,A,b,Aeq,beq)`: 等式约束 $Aeq*x=beq$, 若没有不等式约束 $A*x \leq b$, 则置 $A=[]$, $b=[]$ 。

`x = linprog(f,A,b,Aeq,beq,lb,ub)`: 指定 x 的范围 $lb \leq x \leq ub$, 若没有等式约束 $Aeq*x=beq$, 则置 $Aeq=[]$ 、 $beq=[]$ 。

`x = linprog(f,A,b,Aeq,beq,lb,ub,x0)`: 置初值 x_0 。

`x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)`: `options` 为指定的优化参数。

`[x,fval] = linprog(...)`: 返回目标函数最优值, 即 $fval=c'*x$ 。

`[x,lambda,exitflag] = linprog(...)`: `lambda` 为解 x 的拉格朗日乘子。

`[x,lambda,exitflag,output] = linprog(...)`: `exitflag` 为终止迭代的错误条件。

`[x,fval,exitflag,output,lambda] = linprog(...)`: `output` 为关于优化的一些信息。

在 `linprog` 函数中, `exitflag` 可能的退出标志及相关含义描述见表 10-1。在这些调用格式中, 如果线性规划问题中不存在不等式约束, 则设置 $A=[]$ 、 $b=[]$; 如果不存在等式约束时, 则设置 $Aeq=[]$ 、 $beq=[]$; 决策变量没有上下限的约束时, 则设置 $lb=[]$ 、 $ub=[]$ 。对于 `linprog` 函数, 在 `options` 设置中, 可以设置 `LargeScale` 为 `off` 时, 使用单纯形 Simplex 算法对问题进行求解。

表 10-1 `linprog` 函数 `exitflag` 标志及含义

exitflag 标志	含 义
1	函数收敛到最优解
0	达到最大的函数计算次数或迭代次数
-2	没有找到可行解
-3	求解问题无边界约束
-4	算法求解过程中遇到非实值 NaN
-5	原问题和对偶问题均不可行
-7	搜索方向太小

【例 10-3】求解线性规划最小化问题。

$$\begin{aligned} \min f(x) &= -5x_1 - 4x_2 - 6x_3 \\ s.t. \begin{cases} x_1 - x_2 + x_3 \leq 20 \\ 3x_1 + 2x_2 + 4x_3 \leq 42 \\ 3x_1 + 2x_2 \leq 30 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases} \end{aligned}$$

其实现的 MATLAB 程序代码如下:

```
>> clear all;
f = [-5; -4; -6];      % 目标函数系数向量
% 不等式约束系数矩阵
A = [1  -1  1
     3   2  4
     3   2  0];
```

```
b = [20; 42; 30];           %不等式约束常数项向量
lb = zeros(3,1);           %优化变量的下限
% 设置 linprog 函数为标准算法,显示迭代过程,采用 simplex 算法
options=optimset('simplex','on','Display','iter','Largescale','off');
[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb,[],[],options)
```

运行程序, 输出如下:

The default starting point is feasible, skipping Phase 1.

Phase 2: Minimize using simplex.

Iter	Objective f*x	Dual Infeasibility A'*y+z-w-f
0	0	8.774 96
1	-63	1.118 03
2	-78	0

Optimization terminated.

x =

0
15
3

fval = -78

exitflag = 1

output =

iterations: 2

algorithm: 'medium scale: simplex'

cgiterations: []

message: 'Optimization terminated.'

lambda =

ineqlin: [3x1 double]

eqlin: [0x1 double]

upper: [3x1 double]

lower: [3x1 double]

【例 10-4】市场上有 n 种资产（股票、债券……） $S_i (i=1,2,\dots,n)$ 供投资者选择，某公司有数额为 M 的相当大的一笔资金可用做这一个时期的投资。公司财务分析人员对 S_i 种资产进行评估，估算在这一时期内购买 S_i 的平均收益率为 r_i ，并预测出购买 S_i 的损失率为 q_i ，考虑到投资越分散，总的风险就越小，公司已确定当用这笔资金购买若干种资产时，总体风险可用所投资的 S_i 中的最大一个风险来度量，购买 S_i 要付交易费，费率为 p_i ，并且当购买额不超过给定值 u_i 时，交易费按购买 u_i 计算（不买当然无须付费）。另外，假定同期银行利率是 r_0 ，且既无交易费又无风险（ $r_0=5\%$ ）。 $n=4$ 时的相关数据见表 10-2。

表 10-2 不同资产的收益率、损失率、交易费率与限额

S_i	$r_i/\%$	$q_i/\%$	$p_i/\%$	u_i
S_1	28	2.5	1	103
S_2	21	1.5	2	198
S_3	23	5.5	4.5	52
S_4	25	2.6	6.5	40

试给该公司设计一种投资组合方案,即用给定的资金 M ,有选择地购买若干种资产或存银行生息,使净收益尽可能大,而总体风险尽可能小。

本题可建立一个确定投资比例向量模型,使资产组合净收益尽可能大,而总风险尽可能小。

建立模型: 设 x_0, x_1, x_2, x_3, x_4 分别是银行存款和投资于 S_1, S_2, S_3, S_4 的投资比例系数, 由于银行存款既无交易费又没有风险, 故 $p_0 = 0, q_0 = 0$, 总体风险可用所投资的 S_i 中最大的一个风险来度量, 于是投资组合总体风险为

$$F = \max_{0 \leq i \leq 4} \{x_i q_i\}$$

由于题设给出 M 为相当大的一笔资金, 为了简化模型, 可认为该公司购买每一项资产都超过给定的定值 u_i , 于是资金组合的平均收益率为

$$R = \sum_{i=0}^4 x_i (r_i - p_i)$$

为了使得平均收益率尽可能大, 而总体风险尽可能小, 可采用固定总体风险的一个上界 q 使得总体收益取得最大, 为此建立如下的线性规划模型:

$$\begin{aligned} \max R &= \sum_{i=0}^4 x_i (r_i - p_i) \\ \text{s.t.} \quad &\begin{cases} x_0 + x_1 + x_2 + x_3 + x_4 = 1 \\ x_i q_i \leq q \quad (i = 0, 1, 2, 3, 4) \\ x_0, x_1, x_2, x_3, x_4 \geq 0 \end{cases} \end{aligned}$$

模型的求解: 对总体风险的上界从 $[0, 3]$, 取步长为 0.01, 计算 301 种不同风险时的总体收益的最大值及相应的投资比例系数, 并给出投资方案的净收益与风险损失率的关系图。

其实现的 MATLAB 程序代码如下:

```
>> clear all;
C=[-27 -19 -18.5 -18.5 -5];
a=[2.5 0 0 0 0;0 1.5 0 0 0;0 0 5.5 0 0;0 0 0 2.6 0;0 0 0 0 0];
a1=[1 1 1 1 1];
b1=1;
lb=[0;0;0;0;0];
t=0:0.01:3;
b=t(ones(5,1),:);
for k=1:301;
    [x(:,k),y(k)]=linprog(C,a,b(:,k),a1,b1,lb);
end
plot(t,-y); %绘出投资方案的净收益率与风险损失率的关系图
grid on;
h1=polyfit(t(1:62),-y(1:62),1) %拟合净收益率与风险损失率的关系
h2=polyfit(t(62:251),-y(62:251),1)
h3=polyfit(t(251:301),-y(251:301),1)
%输出 26 种风险时的各种资产的投资比例系数与收益矩阵
tz=[t(1:10:251)' [x(:,1:10:251)' -y(1:10:251)']]
```

运行程序, 输出如下:

```

h1 =
    25.7802    5.0000
h2 =
    3.2441   18.9156
h3 =
    0.0000   27.0000
tz =
    0         0.0000    0.0000    0.0000    0.0000    1.0000    5.0000
    0.1000    0.0400    0.0667    0.0182    0.0385    0.8367    7.5780
    0.2000    0.0800    0.1333    0.0364    0.0769    0.6734   10.1560
    0.3000    0.1200    0.2000    0.0545    0.1154    0.5101   12.7341
    0.4000    0.1600    0.2667    0.0727    0.1538    0.3468   15.3121
    0.5000    0.2000    0.3333    0.0909    0.1923    0.1834   17.8901
    0.6000    0.2400    0.4000    0.1091    0.2308    0.0201   20.4681
    0.7000    0.2800    0.4667    0.1222    0.1312    0.0000   21.1133
    0.8000    0.3200    0.5333    0.0752    0.0714    0.0000   21.4867
    0.9000    0.3600    0.6000    0.0120    0.0280    0.0000   21.8600
    1.0000    0.4000    0.6000    0.0000    0.0000    0.0000   22.2000
    1.1000    0.4400    0.5600    0.0000    0.0000    0.0000   22.5200
    1.2000    0.4800    0.5200    0.0000    0.0000    0.0000   22.8400
    1.3000    0.5200    0.4800    0.0000    0.0000    0.0000   23.1600
    1.4000    0.5600    0.4400    0.0000    0.0000    0.0000   23.4800
    1.5000    0.6000    0.4000    0.0000    0.0000    0.0000   23.8000
    1.6000    0.6400    0.3600    0.0000    0.0000    0.0000   24.1200
    1.7000    0.6800    0.3200    0.0000    0.0000    0.0000   24.4400
    1.8000    0.7200    0.2800    0.0000    0.0000    0.0000   24.7600
    1.9000    0.7600    0.2400    0.0000    0.0000    0.0000   25.0800
    2.0000    0.8000    0.2000    0.0000    0.0000    0.0000   25.4000
    2.1000    0.8400    0.1600    0.0000    0.0000    0.0000   25.7200
    2.2000    0.8800    0.1200    0.0000    0.0000    0.0000   26.0400
    2.3000    0.9200    0.0800    0.0000    0.0000    0.0000   26.3600
    2.4000    0.9600    0.0400    0.0000    0.0000    0.0000   26.6800
    2.5000    1.0000    0.0000    0.0000    0.0000    0.0000   27.0000
    
```

投资方案的净收益与风险损失率的拟合方程如下：

$$R = \begin{cases} 25.7802q + 5.00 & (0 \leq q \leq 0.62) \\ 3.2441q + 18.9156 & (0.62 \leq q \leq 2.5) \\ 27 & (q \geq 2.5) \end{cases}$$

其拟合图形效果如图 10-1 所示。

从图中可见，投资方案的净收益率与风险损失率关系呈 3 段折线：在风险损失率增长的初期净收益增长得很快，但是当风险达到一定的时候其增长趋缓，当风险达到 2.5 时投资方案就只有一个，即只购买风险最大且收益也最大的一种资产，此时投资收益与风险之间的关系为一条水平直线。

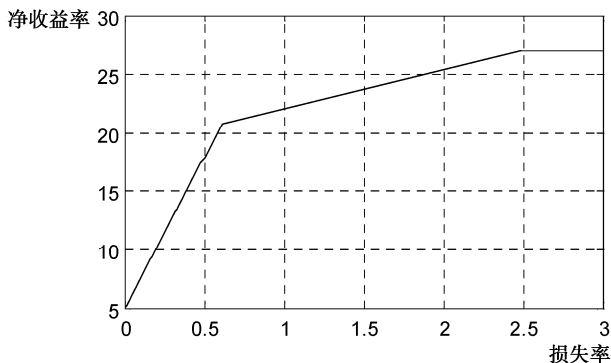


图 10-1 投资方案的净收益率与风险损失率的关系图

10.3 非线性优化问题

10.3.1 有约束优化问题

在工程实际中，所建立的数学模型通常都包含各种不同的约束条件，可能是线性等式、不等式约束，也可能是非线性等式、不等式约束。对于有约束问题，可以描述如下：

- (1) 目标函数： $\min f(x)$ 。
- (2) 线性不等式约束： $A \cdot x \leq b$ 。
- (3) 线性等式约束： $A_{eq} \cdot x = b_{eq}$ 。
- (4) 非线性不等式约束： $c(x) \leq 0$ 。
- (5) 非线性等式约束： $c_{eq}(x) = 0$ 。
- (6) 变量上下限约束： $lb \leq x \leq ub$ 。

MATLAB 优化工具箱中，使用 `fmincon` 函数对有约束问题进行求解，其调用格式如下：

```
x = fmincon(fun,x0,A,b)
x = fmincon(fun,x0,A,b,Aeq,beq)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
[x,fval] = fmincon(...)
[x,fval,exitflag] = fmincon(...)
[x,fval,exitflag,output] = fmincon(...)
[x,fval,exitflag,output,lambda] = fmincon(...)
[x,fval,exitflag,output,lambda,grad] = fmincon(...)
[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)
```

其中，`fun` 为目标函数，`x0` 为初始值，`A`、`b` 满足线性不等式约束 $Ax \leq b$ ，若没有不等式约束，则取 `A=[]`，`b=[]`；`Aeq`、`beq` 满足等式约束 $Aeqx = beq$ ，若不满足，则取 `Aeq=[]`，`beq=[]`；`lb`、`ub` 满足 $lb \leq x \leq ub$ ，若没有界，可设 `lb=[]`，`ub=[]`；`lambda` 是拉格朗日乘子，它体现哪一个约束有效；`output` 输出优化信息；`grad` 表示目标函数在 `x` 处的梯度。

`fmincon` 函数退出标志 `exitflag` 及相应的含义见表 10-3。

表 10-3 fmincon 函数 exitflag 退出标志及相应含义

exitflag 标志	含 义
1	满足优化设定误差容忍度
2	优化变量的变化量小于给定的容差 TolX 属性值
3	目标函数的变化量小于给定的容差 TolFun 属性值
4	搜索方向幅值小于给定容差或约束违背小于约束容差 TolCon
5	搜索方向变化率小于给定容差或约束违背小于约束容差 TolCon
0	达到最大的函数计算次数或者迭代次数
-1	由于调用输出函数而终止优化计算
-2	没有找到可行解

【例 10-5】求解有约束优化问题。

$$\begin{aligned} \min f(x) &= x^2 + y^2 \\ s.t. \begin{cases} x^2 + y^2 \leq 5 \\ x_1 + 2x_2 = 4 \\ x \geq 0, y \geq 0 \end{cases} \end{aligned}$$

利用 fmincon 函数来求解优化问题，首先建立目标函数文件：

```
function feval=li10_5A(x)
```

```
feval=x(1)^2+x(2)^2;
```

建立约束条件的 M 文件：

```
function [c,ceq]=li10_5B(x)
```

```
c(1)=x(1)^2+x(2)^2-5;
```

```
ceq=[];
```

其实现的 MATLAB 程序代码如下：

```
>> clear all;
A=[];           %线性不等式系数矩阵为空
b=[];           %线性不等式常数向量
Aeq=[1 2];      %等式约束系数矩阵
beq=[4];        %等式约束右侧向量
lb=[0,0];       %优化变量的下限约束
ub=[];          %优化变量的上限约束
x0=[0;0];       %优化变量的初始迭代点
%迭代次数显示,最大函数计算次数设置
options=optimset('Display','iter','MaxFunEvals',1e5);
[x,fval,exitflag,output,lambda,grad,hessian]=fmincon...
('li10_5A',x0,A,b,Aeq,beq,lb,ub,'li10_5B',options)
```

运行程序，输出如下：

```

Max      Line search  Directional  First-order
Iter F-count      f(x)   constraint  steplength  derivative  optimality Procedure
    0           3          0           0           0           4
Infeasible start point
```

```

1      6      3.2      0      1      2e-008      4.8
x =      %最优解和对应目标函数值
0.8000
1.6000
fval =
3.2000
% 退出标志, 表明 fmincon 函数求解问题收敛于最优解
exitflag =      1
output =      %输出信息
iterations: 2
funcCount: 6
lssteplength: 1
stepsize: 2.8988e-008
algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
firstorderopt: 2.5928e-008
constrviolation: 0
message: [1x834 char]
lambda =      %约束条件的信息输出
lower: [2x1 double]
upper: [2x1 double]
eqlin: 1.6000
eqnonlin: [0x1 double]
ineqlin: [0x1 double]
ineqnonlin: 0
grad =      %目标函数在最优解处的梯度值
1.6000
3.2000
hessian =      %目标函数在最优解处的 Hessian 矩阵
1.2000      0.4000
0.4000      1.8000

```

在 MATLAB 的最优化工具箱中, 提供了 `fminbnd` 函数用于求解只含优化变量上下限约束的有约束问题, 即

$$\min_x f(x)$$

`fminbnd` 函数用于求解只含优化变量约束的连续性问题, 而且有可能只得到局部最优解。当最优解出现在约束边界上时, `fminbnd` 函数收敛速度慢, 此时可以使用 `fmincon` 函数来加快收敛速度, `fminbnd` 函数的调用格式如下:

```

x = fminbnd(fun,x1,x2)
x = fminbnd(fun,x1,x2,options)
[x,fval] = fminbnd(...)
[x,fval,exitflag] = fminbnd(...)
[x,fval,exitflag,output] = fminbnd(...)

```

同样, 返回变量 `exitflag` 的不同值表示了 `fminbnd` 函数求解问题的不同状态。表 10-4 所示为 `fminbnd` 函数退出标示 `exitflag` 取值及相应含义描述, 通过返回状态 `exitflag`, 可以判断 `fminbnd` 函数求解边界约束问题的成功与否。

表 10-4 fminbnd 函数退出标志 exitflag 取值及含义

exitflag 标志	含 义
1	目标函数收敛于最优解
0	目标函数最大计算次数超过 MaxFunEvals 属性值或者最大迭代次数 MaxIter
-1	算法由于输出函数而终止
-2	约束边界不一致

【例 10-6】求解二次函数

$$\min f(x) = (x - a)^2 + b$$

已知 $a = 5$ 、 $b = 3$ ， x 的定义域区间分别为 $x \in [0, 7]$ 和 $x \in [0, 3]$ 。

先建立二次函数的 M 文件：

```
function feval=li10_6A(x,a,b)
```

```
feval=(x-a)^2+b;
```

其实现的 MATLAB 程序代码如下：

```
>> a=5;b=3;
[x,fval,exitflag,output]=fminbnd(@(x) li10_6A(x,a,b),0,7)
```

运行程序，输出如下：

```
x =
    5
fval =
    3
exitflag =
    1
output =
    iterations: 5
    funcCount: 6
    algorithm: 'golden section search, parabolic interpolation'
    message: [1x112 char]
```

从 $\text{exitflag}=1$ 可以看出，fminbnd 函数求解边界约束成功，目标函数收敛于最优解 $x=5$ 。当 $x \in [0, 3]$ 时，只需要再执行如下代码：

```
>> a=5;b=3;
[x,fval,exitflag,output]=fminbnd(@(x) li10_6A(x,a,b),0,3)
```

运行程序，输出如下：

```
x =
    3.0000
fval =
    7.0002
exitflag =
    1
output =
    iterations: 22
    funcCount: 23
    algorithm: 'golden section search, parabolic interpolation'
    message: [1x112 char]
```

【例 10-7】求解二元函数的全局极大值和全局极小值。

$$\min f(x, y) = (x - 10y)^2 + (y - 2x)^4 + 40[\sin^2(10x) + \sin^2(10y)]$$

其中, $x \in [-1, 1]$, $y \in [-1, 1]$ 。

首先建立二元函数的 M 文件:

```
function feval=li10_7A(x)
```

```
feval=(x(1)-10*x(2))^2+(x(2)-2*x(1))^4+40*(sin(10*x(1))^2+sin(10*x(2))^2);
```

其实现的 MATLAB 程序代码如下:

```
>> [x,fval,exitflag,output]=fminbnd('li10_7A',[-1 1],[1 1])
```

运行程序, 输出如下:

```
x =
    -0.2361    1.0000
fval =
    141.1327
exitflag =
     1
output =
    iterations: 0
    funcCount: 1
    algorithm: 'golden section search, parabolic interpolation'
    message: [1x112 char]
```

从 fminbnd 函数退出标志 exitflag 虽然可以看到函数最终收敛于最优解, 但实际上, fminbnd 函数所得到的极值点为一个局部极值点, 该二元函数的全局极值点为 $x^* = 0$, $y^* = 0$ 。为了比较 fminbnd 函数和 fmincon 函数在边界约束上的差异, 使用 fmincon 函数对该二元函数进行求解, 其实现的 MATLAB 程序代码如下:

```
>> x0=[-1;1];
lb=-1*ones(1,2);      %优化变量的下限约束
ub=ones(1,2);         %优化变量的上限约束
[x,fval,exitflag,output,lambd,grad,hessian]=fmincon('li10_7A',x0,[],[],[],[],lb,ub)
```

运行程序, 输出如下 (结果省略 lambd 及 grad 的输出):

```
x =
    1.0e-013 *
    0.5773
    0.4996
fval =
    2.3511e-023
exitflag =
     5
output =
    iterations: 3
    funcCount: 9
    lssteplength: 1
    stepsize: 8.5491e-005
    algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 6.1095e-005
    constrviolation: -1.0000
```

```
message: [1x844 char]
hessian =
    281.1511   -322.0270
   -322.0270    371.1512
```

从 `fmincon` 函数的结果中可以看出，此时求解的最优极值点和全局的极值点误差非常小，可以认为此时求解结果是可信的。

从这个示例可以看出，`fminbnd` 函数在求解包含多个极值点的边界约束最优化问题时，很容易隐含局部最优，而 `fmincon` 函数可以克服这个缺点，能够比较准确地找到全局优点。

10.3.2 无约束优化问题

无约束优化问题是指目标函数不存在约束条件包括上下限幅值约束。无约束优化问题的标准形式为 $\min_x f(x)$ 。

在工程优化中，用于求解无约束优化问题的方法非常多，包括最简单的高等数学微分方法，即优化函数在极值点处的偏微分为零，进而可以求解方程，获取无约束优化问题的极值点。在数值解法中，无约束优化问题的求解包括一维搜索、最速下降法、共轭梯度法、牛顿法和拟牛顿法等。MATLAB 优化工具箱提供了 `fminunc` 函数和 `fminsearch` 函数用于求解无约束优化问题。`fminunc` 函数的调用格式如下：

```
x = fminunc(fun,x0)
x = fminunc(fun,x0,options)
x = fminunc(problem)
[x,fval] = fminunc(...)
[x,fval,exitflag] = fminunc(...)
[x,fval,exitflag,output] = fminunc(...)
[x,fval,exitflag,output,grad] = fminunc(...)
[x,fval,exitflag,output,grad,hessian] = fminunc(...)
```

在 `fminunc` 函数的输出中，包含了 `grad` 和 `hessian` 两个参数。其中 `grad` 表示目标函数在最优解处的梯度，而 `hessian` 参数表示目标函数在最优解处的 Hessian 矩阵。`fminunc` 函数退出标志取值及相关含义见表 10-5。对于 `fminunc` 函数的 `options` 设置，通过设置 `HessUpdate` 属性值来选择拟牛顿法的搜索方法，包括 `bfgs`、`dfp` 和 `stepd` 方法。同时也可以设置 `Gradobj` 属性选项和 `Hessian` 选项来确定是否使用目标函数梯度和 Hessian 矩阵信息。

表 10-5 `fminunc` 函数 `exitflag` 标志取值及含义

exitflag 标志	含 义
1	返回目标函数最优解梯度幅值小于给定容差
2	优化变量 x 的变化小于给定容差 <code>TolX</code> 属性值
3	目标函数值的变化小于给定误差 <code>TolFcn</code> 属性值
0	目标函数最大计算次数超过 <code>MaxFunEvals</code> 属性值或者最大迭代次数 <code>MaxIter</code>
-1	算法由于输出函数而终止
-2	在当前搜索方向上线性搜索无法找到可行解

【例 10-8】求解无约束优化问题。

$$\min f(x) = 3x_1^2 + 2x_1x_2 + x_2^2$$

首先建立目标函数 M 文件:

```
function f = li10_8A(x)
```

```
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;
```

其实现的 MATLAB 程序代码如下:

```
>> x0 = [1,1];
```

```
[x,fval,exitflag,output,grad,hessian] = fminunc(@li10_8A,x0)
```

运行程序, 输出如下 (省略了 output 输出结构体数据):

```
x =
    1.0e-006 *
    0.2541    -0.2029
fval =
    1.3173e-013
exitflag =
     1
% 还包括目标函数在最优解处的梯度和 Hessian 矩阵
grad =
    1.0e-005 *
    0.1163
    0.0087
hessian =
    6.0000    2.0000
    2.0000    2.0000
```

在 fminunc 函数中, 可以设置 options 结构体选项, 考虑目标函数的梯度和 Hessian 矩阵。

首先定义目标函数 M 文件:

```
function [f,g] = li10_8A(x)
```

```
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;    % Cost function
```

```
if nargout > 1
```

```
    g(1) = 6*x(1)+2*x(2);
```

```
    g(2) = 2*x(1)+2*x(2);
```

```
end
```

其实现的 MATLAB 程序代码如下, 同样可以得到相同的计算结果:

```
>> options = optimset('GradObj','on');
```

```
x0 = [1,1];
```

```
[x,fval] = fminunc(@li10_8A,x0,options)
```

fminunc 函数在进行无约束优化问题求解时, 要求判断目标函数在优化变量处的梯度和 Hessian 矩阵, 因此 fminunc 函数仅适用于目标函数为连续的情况, 同时 fminunc 只能用来求解优化变量为实数的问题, 当优化变量为复数时, 需要将问题分解成实部和虚部分别进行无约束优化求解。

除了 fminunc 函数外, MATLAB 工具箱还提供了 fminsearch 函数进行无约束优化问题的求解, 与 fminunc 函数相比, fminsearch 函数可以用来求解目标函数进行无约束优化问题的连续, 在最优解附件出现奇异值等问题, 只能给出局部最优解, 同样地, fminsearch 函数只能求解实数最优化问题。

fminsearch 函数的调用格式如下：

`x = fminsearch(fun,x0)`

`x = fminsearch(fun,x0,options)`

`[x,fval] = fminsearch(...)`

`[x,fval,exitflag] = fminsearch(...)`

`[x,fval,exitflag,output] = fminsearch(...)`

在 fminsearch 函数中，输出标志 exitflag 以及对应的含义见表 10-6。

表 10-6 fminsearch 函数输出 exitflag 标志及含义

exitflag 标志	含 义
1	目标函数收敛于最优解
0	目标函数最大计算次数超过 MaxFunEvals 属性值或者最大迭代次数 MaxIter
-1	算法由于输出函数而终止

【例 10-9】求解无约束优化问题。

$$\min f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

首先定义目标函数 M 文件：

`function f = li10_9A(x)`

`f = 100*(x(2)-x(1)^2)^2+(1-x(1))^2;`

其实现的 MATLAB 程序代码如下：

```
>> x0=[0;1];
```

```
[x,fval,exitflag]=fminsearch('li10_9A',x0)
```

运行程序，输出如下：

```
x =
    1.0000
    1.0000
fval =
    2.4289e-010
exitflag =
    1
```

对于目标函数的建立，除了通过 M 文件函数外，还可以通过以下三种方法来进行定义。

(1) 用函数句柄@来进行函数定义：

```
li10_9B=@(x) 100*(x(2)-x(1)^2)^2+(1-x(1))^2
```

```
x0=[0;1];
```

```
[x,fval,exitflag] = fminsearch(li10_9B,x0)
```

(2) 定义函数字符串：

```
li10_9C='100*(x(2)-x(1)^2)^2+(1-x(1))^2';
```

```
x0=[0;1];
```

```
[x,fval,exitflag] = fminsearch(li10_9C,x0)
```

(3) 定义 inline 函数：

```
li10_9D=inline('100*(x(2)-x(1)^2)^2+(1-x(1))^2');
```

```
x0=[0;1];
```

```
[x,fval,exitflag] = fminsearch(li10_9D,x0)
```

如果在目标函数中出现外部输入变量，可以通过以下两种方法对目标函数进行定义。

(1) 使用函数柄@来进行含有参数的函数定义：

```
li10_9E = @(x)100*(x(2)-x(1)^2)^2+(a-x(1))^2;
```

```
a=1.5;
```

```
x0=[0;1]
```

```
[x,fval,exitflag] = fminsearch(li10_9E,x0)
```

(2) 定义含有参数的 M 文件：

```
function f = li10_9F(x,a)
```

```
f = x(1)^2 + a*x(2)^2;
```

```
a=1.5;
```

```
x0=[0;1]
```

```
[x,fval,exitflag] = fminsearch(@(x) li10_9F(x,a),x0)
```

下面通过一个示例来说明线性优化在实例中的应用。

【例 10-10】某公司欲以每件 2 元的价格购进一批商品。一般来说随着商品售价的提高，预期销售量将减少，对此进行估算，结果见表 10-7 的一、二栏。为了尽快回收资金并获得较多的赢利，公司打算做广告，投入一定的广告费后，销售量将有一个增长，可由销售增长因子来表示。据统计，广告费与销售增长因子关系见表 10-7 的三、四栏所列。问公司采取怎样的营销决策能使预期的利润最大？

表 10-7 售价与预期销售量、广告费与销售增长因子

售价/元	2.00	2.50	3.00	3.50	4.00	4.50	5.00	5.50	6.00
预期销售量/万元	4.1	3.8	3.4	3.2	2.9	2.8	2.5	2.2	2.0
广告费/万元	0	1	2	3	4	5	6	7	
销售增长因子	1.00	1.40	1.70	1.85	1.95	2.00	1.95	1.80	

分析：设 x 表示售价（单位：元）， y 表示预期销售量（单位：万元）， z 表示广告费（单位：万元）， k 表示销售增长因子。投入广告费后，实际销售量记为 s （万元），获得的利润记为 p （单位：万元）。由表 10-7 易见预期销售量 y 随着售价 x 的增加而单调下降，而销售增长因子 k 在开始时随着广告费 z 的增加而增加，在广告费 z 等于 5 万元时达到最大值，然后在广告费增加时反而有所回落，为此先画出散点图。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x=[2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0];
s=[4.1 3.8 3.4 3.2 2.9 2.8 2.5 2.2 2.0];
figure(1);
plot(x,'s','r-p') %画售价与预期销售量散点图（10-2（a））
z=[0,1,2,3,4,5,6,7];
k=[1.00 1.40 1.70 1.85 1.95 2.00 1.95 1.80];
figure(2);plot(z,k,'r-p'); %画出广告费与销售增长因子散点图（10-2（b））
```

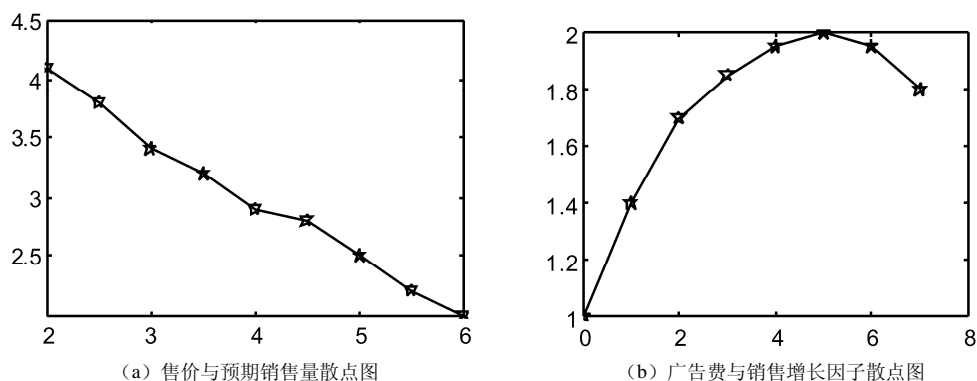


图 10-2 散点图

从图 10-2 易见，售价 x 与预期销售量 y 近似于一条直线，广告费 z 与销售增长因子 k 近似于一条二次曲线，为此建立拟合函数模型，令

$$\begin{cases} y = ax + b \\ k = c + dz + ez^2 \end{cases}$$

其中系数 a 、 b 、 c 、 d 、 e 是待定参数。

再建立优化模型

$$\begin{aligned} \max_{x,z} p &= (c + dz + ez^2)(a + bx)(x - 2) - z \\ \text{s.t.} \quad &\begin{cases} x > 0 \\ z > 0 \end{cases} \end{aligned}$$

先求拟合函数的系数 a 、 b 、 c 、 d 、 e ，并画出散点图和拟合曲线，程序命令（接上面的程序）如下：

```
>> a1=polyfit(x,s,1)
a2=polyfit(z,k,2)
```

运行结果如下：

```
a1 =
    -0.5133    5.0422
a2 =
    -0.0426    0.4092
```

即拟合函数的系数 $a=-0.5133$ ， $b=5.0422$ ， $c=1.0188$ ， $d=0.4092$ ， $e=-0.0426$ 。

其次求解优化模型，因 MATLAB 中仅能求极小值，其实现的程序代码如下：

```
function y=nline(x)
y(2)-(-0.5133*(1)+5.0422)*(-0.0423*x(2)^2+0.4092*x(2)+1.0188)*(x(1)-2);
```

在命令窗口中输入：

```
>> [x,fval]=fmincon('nline',[5;3.3],[],[],[],[],[0;0],[]) %求解规划问题
```

输出如下：

```
x = 1.0e+013 *
    1.1504
    0.0000
fval =
```

-1.0449e+014

即当销售价格为 $x=5.9115$ 元, 广告费 $z=3.083$ 万元时, 公司预期的利润最大为 11.6631 万元。

10.4 二次规划问题

二次规划问题在 MATLAB 优化工具箱中可以表示为

$$\min_x \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{f}^T \mathbf{x}$$

$$s.t. \begin{cases} \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \\ \mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq} \\ \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub} \end{cases}$$

标准的二次规划问题形式要求目标函数是关于优化变量的二次函数, 而约束条件只包含线性等式不等式约束和优化变量自身的上下限约束。对于二次规划问题, 如果将目标函数看做非线性函数, 那么二次规划问题则可以使用 `fmincon` 函数来进行求解。在 MATLAB 优化工具箱中, 提供了 `quadprog` 函数, 求解具有标准二次规划形式的问题。`quadprog` 函数的调用格式如下:

```
x = quadprog(H,f,A,b)
x = quadprog(H,f,A,b,Aeq,beq)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
x = quadprog(problem)
[x,fval] = quadprog(...)
[x,fval,exitflag] = quadprog(...)
[x,fval,exitflag,output] = quadprog(...)
[x,fval,exitflag,output,lambda] = quadprog(...)
```

说明: 在 `quadprog` 函数输入变量中, \mathbf{H} 和 \mathbf{f} 分别为二次项系数矩阵和一次项系数向量。在对二次规划问题求解时, 首先需要将目标函数转化为二次规划问题标准形式, 得到系数矩阵 \mathbf{H} 和系数向量 \mathbf{f} 。`quadprog` 函数退出标志 `exitflag` 及其相应含义见表 10-8。

表 10-8 `quadprog` 函数退出标志 `exitflag` 及含义

exitflag 标志	含 义
1	目标函数收敛于最优解
3	目标函数的变化小于给定的容差 TolFcns
4	找到局部极小值点
0	迭代次数超过最大迭代次数 MaxIter
-2	求解问题无可行解
-3	求解问题无边界
-4	当前搜索方向不是下降方向, 搜索过程结束
-7	搜索方向幅值太小, 搜索过程结束

【例 10-11】求解一次规划问题。

$$\min f(x) = \frac{1}{2}x_1^2 + x_2^2 - x_1x_2 - 2x_1 - 6x_2$$

$$s.t. \begin{cases} x_1 + x_2 \leq 2 \\ -x_1 + 2x_2 \leq 2 \\ 2x_1 + x_2 \leq 3 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

对于二次规则问题，首先必须将目标函数转化为标准形式，得到二次项系数矩阵 \mathbf{H} 和一次项系数向量 \mathbf{f} 。通过分析，可以得到 \mathbf{H} 和 \mathbf{f} 分别为

$$\mathbf{H} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}, \quad \mathbf{f} = (-2, -6)^T$$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
H = [1 -1; -1 2];
f = [-2; -6];
A = [1 1; -1 2; 2 1];
b = [2; 2; 3];
lb = zeros(2,1);
```

运行程序，输出如下：

```
x =
    0.6667
    1.3333
fval =
   -8.2222
exitflag =
     1
output =
    iterations: 3
    algorithm: 'medium-scale: active-set'
firstorderopt: []
    cgiterations: []
    message: 'Optimization terminated.'
```

退出标志 `exitflag` 表明函数收敛于极值点。同样，将目标函数看做是非线性函数时，可以使用 `fmincon` 函数对二次规划问题进行求解。其实现的 MATLAB 程序代码如下：

```
>> clear all;
A = [1 1; -1 2; 2 1];
b = [2; 2; 3];
lb = zeros(2,1);
x0=[0;1];
li10_11=@(x)(1/2*x(1)^2+x(2)^2-x(1)*x(2)-6*x(2)); % @函数句柄
options=optimset('Display','iter','MaxFunEvals',1e5);
[x,fval,exitflag,output,lambda,grad,hessian]=fmincon(li10_11,x0,A,b,[],[],lb,[],[],options)
```

运行程序，显示的迭代过程如下：

Iter	F-count	f(x)	constraint	Max steplength	Line search derivative	Directional optimality	First-order Procedure
0	3	-5	0				
1	6	-6.88889	0	1	-2.68		0.333

经过2次迭代后,方向导数为-2.68,因此搜索方向导数过小且为负导致搜索过程结束,退出标志 `exitflag=1`。输出结果如下:

```
x =
    0.6667
    1.3333
fval =
   -6.8889
exitflag =
     1
output =
    iterations: 2
    funcCount: 6
    lssteplength: 1
    stepsize: 0
    algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
    firstorderopt: 0
    constrviolation: 0
    message: [1x834 char]
```

由于 `fmincon` 函数退出标志 `exitflag=1`, 因此可以判断目标函数收敛于最优解, 比较 `fmincon` 和 `quadprog` 函数的求解结果, 可以看出, `fmincon` 函数的求解结果比 `quadprog` 函数要好。

【例 10-12】求解二次规划问题。

$$\begin{aligned} \min & 2x_1^2 + x_2^2 + x_3^2 - x_1x_2 \\ \text{s.t.} & \begin{cases} x_1 + x_2 \leq 4 \\ 5x_1 + x_3 = 10 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases} \end{aligned}$$

该二次规划问题中, 约束条件同时包含了线性等式约束和线性不等式约束。首先同样需要将目标函数转化为标准形式, 得到系数矩阵 \mathbf{H} 和向量 \mathbf{f} , 通过变换, 得

$$\mathbf{H} = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

使用 `quadprog` 函数求解二次规划问题的 MATLAB 程序代码如下:

```
>> clear all;
H=[4 -1 0;-1 2 0;0 0 2]; %二次规划标准变换后的 H 矩阵
f=zeros(3,1); %二次规划标准变换后的 f 矩阵
lb=zeros(3,1); %优化变量的下限约束
A=[1 1 0]; %线性不等式的系数矩阵
b=4; %线性不等式右侧常数向量
Aeq=[5 0 1]; %等式约束的系数矩阵
```

```
beq=[10]; %等式约束右侧常数向量
[x,fval,exitflag,output,lambda]=quadprog(H,f,A,b,Aeq,beq,lb)
```

运行程序，输出如下：

```
x =
    1.8692
    0.9346
    0.6542
fval =
    6.5421
exitflag =
     1
output =
    iterations: 1
    algorithm: 'medium-scale: active-set'
    firstorderopt: []
    cgiterations: []
    message: 'Optimization terminated.'
```

使用 `fmincon` 函数同样可以对二次规划问题进行求解，其实现的 MATLAB 程序代码如下：

```
>> A=[1 1 0];
b=4;
Aeq=[5 0 1];
beq=[10];
lb=zeros(3,1);
x0=[1;1;1];
li10_12=@(x)(2*x(1)^2-x(1)*x(2)+x(2)^2+x(3)^2); % @函数
options=optimset('Display','iter','MaxFunEvals',1e5); % 优化过程属性选项的设置
[x,fval,exitflag,output,lambda,grad,hessian]=fmincon(li10_12,x0,A,b,Aeq,beq,lb,[],[],options)
```

运行程序，显示的迭代过程如下：

Iter	F-count	f(x)	Max constraint	Line search steplength	Directional derivative	First-order optimality	Procedure
0	4	3	4				Infeasible start point
1	8	8	0	1	1.72e-008	12	
2	12	6.56296	0	1	-2.54	12.2	
3	16	6.5421	0	1	-0.305	13.1	
4	20	6.54209	1.776e-015	1	-0.0134	13.1	
5	24	6.54206	0	1	-0.0116	13.1	Hessian modified

输出结果如下：

```
x =
    1.8692
    0.9346
    0.6542
fval =
    6.5421
exitflag =
     4
output =
    iterations: 6
```



```

funcCount: 24
lssteplength: 1
stepsize: 8.0493e-007
algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
firstorderopt: 1.3113e-006
constrviolation: 0
message: [1x817 char]

```

对比 quadprog 函数求解结果与 fmincon 函数的求解结果,可以看出两个函数所示二次规划问题的求解结果相同。

10.5 0-1 整数规划问题

整数规划问题就是决策变量为整数的优化问题,如果当决策变量取 0 或 1 时,则对应于 0-1 规划问题,如旅行员最短路径问题(TSP 问题)、背包问题(Knapsack 问题)等。

10.5.1 0-1 整数规划概述

1. 0-1 型整数线性规划的提法

0-1 型整数线性规划是一类特殊的整数规划,它的变量仅取值 0 或 1。它的提法如下:

$$\begin{aligned} \min f &= \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad &\begin{cases} \mathbf{Ax} = \mathbf{b} \\ x_j \quad (j=1,2,\dots,n) \text{ 取0或1} \end{cases} \end{aligned}$$

其中, $\mathbf{c} = (c_1, c_2, \dots, c_n)^T$, $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$, $\mathbf{A} = (a_{ij})_{m \times n}$, $\mathbf{b} = (b_1, b_2, \dots, b_m)^T$ 。

称此时的决策变量为 0-1 变量,或称二进制变量。在实际问题中,如果引进 0-1 变量,就可以把各种需要分别讨论的线性(或非线性)规划问题统一在一个问题中讨论了。

2. 求解 0-1 型整数线性规划的隐枚举法

求解整数线性规划的分支定界法也是一种隐枚举法,0-1 规划可以通过增加限定 $0 \leq x_i \leq 1$ 的整数规划求解。

在此主要介绍一种针对 0-1 型整数规划特点的隐枚举法算法。隐枚举是一种“聪明”的枚举,通过设计一些方法,检查变量是组合的一部分,而不必全部检查 n 个变量的 2^n 个取值组合。要说明的是,对有些问题(特别是对于一部分决策变量是 0-1 变量的混合线性规划中)隐枚举法有时难以适用,所以穷举法还是必要的。

3. 隐枚举原理与算法步骤

(1) 记 $f_0 = \infty$, 将 n 个决策变量构成的 \mathbf{x} 的可能的 2^n 种取值组合按二进制(或某种顺序)排序。

(2) 按上述顺序对 \mathbf{x} 的取值首先检测 $f = \mathbf{c}^T \mathbf{x} \leq f_0$ 是否成立,若不成立则放弃该取值的 \mathbf{x} ,按次序换(1)中下一组 \mathbf{x} 的取值重复上述过程;若成立,则转下一步。

(3) 对 \mathbf{x} 逐一检测 $\mathbf{Ax} \leq \mathbf{b}$ 中的 m 个条件是否满足,一旦检测某条件不满足便停止检测后面的条件,而放弃这一组 \mathbf{x} ,按次序换(1)中下一组 \mathbf{x} 的取值执行(2);若 m 个条件全满足,则转下一步。

(4) 记 $f_0 = \min(f_0, f)$, 按次序转(1)中下一组 \mathbf{x} 的取值,执行(2)。

(5) 最后一组满足 $f = c^T \leq f_0$ 和 $Ax \leq b$ 的 x 即为最优解。

注意：在执行上述算法步骤时，可以及时地记录所有满足 $f = c^T x \leq f^*$ (f^* 为最优值) 的 x ，以便求所有最优解。

10.5.2 0-1 整数规划的实现

在 MATLAB 中提供了 bintprog 函数实现 0-1 型线性规划函数。其调用格式如下：

```
x = bintprog(f)
x = bintprog(f,A,b)
x = bintprog(f,A,b,Aeq,beq)
x = bintprog(f,A,b,Aeq,beq,x0)
x = bintprog(f,A,b,Aeq,Beq,x0,options)
x = bintprog(problem)
[x,fval] = bintprog(...)
[x,fval,exitflag] = bintprog(...)
[x,fval,exitflag,output] = bintprog(...)
```

bintprog 函数的退出标志 exitflag 及相应的含义描述见表 10-9。

表 10-9 bintprong 函数 exitflag 标志及含义

exitflag 标志	含 义
1	目标函数收敛于最优解
0	迭代次数超过最大迭代次数 MaxIter
-2	无可行解
-4	搜索节点数超过最大给定节点数 MaxNodes
-5	搜索时间超过最大给定搜索时间 MaxTime

【例 10-13】求解整数规划问题。

$$\min f(x) = -9x_1 - 5x_2 - 6x_3 - 4x_4$$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
>> f = [-9; -5; -6; -4];
A = [6 3 5 2; 0 0 1 1; -1 0 1 0; 0 -1 0 1];
b = [9; 1; 0; 0];
x = bintprog(f,A,b)
```

运行程序，输出如下：

```
Optimization terminated.
x =
     1
     1
     0
     0
```

【例 10-14】求解下列 0-1 型整数线性规划。

$$\begin{aligned} \max f &= -3x_1 + 2x_2 - 5x_3 \\ \text{s.t.} \quad &\begin{cases} x_1 + 2x_2 - x_3 \leq 2 \\ x_1 + 4x_2 - x_3 \leq 4 \\ x_1 + x_2 \leq 3 \\ 4x_2 + x_3 \leq 6 \\ x_1, x_2, x_3 \text{ 为 } 0 \text{ 或 } 1 \end{cases} \end{aligned}$$

其实现的 MATLAB 程序代码如下：

```
>> clear all;
c=[3,-2,5]; %转换求 max 为求 min
a=[1,2,-1;1,4,1;1,1,0;0,4,1];
b=[2;4;3;6];
x0=bintprog(c,a,b,[],[])
x1=BintLp_E(c,a,b)
x2=BintLp_le(c,a,b)
```

运行程序，输出如下：

```
ans =
    0    0    0
    1    1    1
    0    0    0
```

即得问题的解。

10.6 最大最小化问题

最大最小化问题可以描述为以下标准形式：

$$\begin{aligned} \min_x \max_{F_i} \{F_i(x)\} \\ \text{s.t.} \quad &\begin{cases} c(x) \leq 0 \\ c_{\text{eq}}(x) = 0 \\ \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \\ \mathbf{A}_{\text{eq}} \cdot \mathbf{x} = \mathbf{b}_{\text{eq}} \\ \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub} \end{cases} \end{aligned}$$

对于目标函数 $\min_x \max_{F_i} \{F_i(x)\}$ ，其含义表示对于一组目标函数，确定这些目标函数中最大者，然后将该目标函数对优化变量 x 确定其最小值。在 MATLAB 优化工具箱中，提供了 `fminimax` 函数用于求解最小最大化问题。其调用格式如下：

```
x = fminimax(fun,x0)
x = fminimax(fun,x0,A,b)
x = fminimax(fun,x,A,b,Aeq,beq)
x = fminimax(fun,x,A,b,Aeq,beq,lb,ub)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

```
x = fminimax(problem)
[x,fval] = fminimax(...)
[x,fval,maxfval] = fminimax(...)
[x,fval,maxfval,exitflag] = fminimax(...)
[x,fval,maxfval,exitflag,output] = fminimax(...)
[x,fval,maxfval,exitflag,output,lambda] = fminimax(...)
```

fminimax 函数的退出标志 exitflag 的取值及相应的含义见表 10-10。

表 10-10 fminimax 函数 exitflag 标志及含义

exitflag 标志	含 义
1	目标函数收敛于最优解
4	搜索方向幅值小于给定容差或者约束违背小于给定容差 TolCon
5	方向导数幅值小于给定容差或者约束违背小于给定容差 TolCon
0	迭代次数超过最大迭代次数 MaxIter 或者函数计算次数超过给定的 MaxFunEvals
-1	算法因为输出函数终止
-2	无可行解

【例 10-15】求解最小最大化问题。

$$[f_1(x), f_2(x), f_3(x), f_4(x), f_5(x)]$$

$$s.t. \begin{cases} f_1(x) = 2x_1^2 + x_2^2 - 48x_1 - 40x_2 + 304 \\ f_2(x) = -x_1^2 - 3x_2^2 \\ f_3(x) = x_1 + 3x_2 - 18 \\ f_4(x) = -x_1 - x_2 \\ f_5(x) = x_1 + x_2 - 8 \end{cases}$$

首先，建立目标函数族的 M 文件：

```
function f = li10_15(x)
f(1)= 2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+304;    % 目标函数族
f(2)= -x(1)^2 - 3*x(2)^2;
f(3)= x(1) + 3*x(2) -18;
f(4)= -x(1)- x(2);
f(5)= x(1) + x(2) - 8;
```

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x0 = [0.1; 0.1];
options=optimset('MinAbsMax',5);
[x,fval,maxfval,exitflag] = fminimax(@li10_15,x0,[],[],[],[],[],[],options)
```

运行程序，输出如下：

```
x =
    4.9256
    2.0796
fval =
```

```

37.2356 -37.2356 -6.8357 -7.0052 -0.9948
maxfval =
37.2356
exitflag =
5

```

10.7 多元多目标函数优化

10.7.1 “半无限”多元函数优化

“半无限”多元函数优化问题在 MATLAB 优化工具箱中描述为以下的标准形式：

$$\begin{aligned}
 & \min_x f(x) \\
 & \begin{cases} c(x) \leq 0 \\ c_{\text{eq}}(x) = 0 \\ A \cdot x \leq b \\ A_{\text{eq}} \cdot x \leq b_{\text{eq}} \\ \text{s.t.} \quad lb \leq x \leq ub \\ K_1(w, x) \leq 0 \\ K_2(w, x) \leq 0 \\ \dots \\ K_n(w, x) \leq 0 \end{cases}
 \end{aligned}$$

式中 $c(x)$ 和 $c_{\text{eq}}(x)$ 表示非线性不等式和等式约束； $A \cdot x \leq b$ 和 $A_{\text{eq}} \cdot x \leq b_{\text{eq}}$ 表示线性不等式和等式约束； $K_i(w, x)$ 为关于优化变量 x 和变量 w 的函数关系， w 为长度大于 2 的向量，即 $w = (w_1, w_2, \dots, w_n)$ ， $n \geq 2$ 。

在 MATLAB 优化工具箱中，使用 `fseminf` 函数求解“半无限”多元函数优化问题，其调用格式如下：

```

x = fseminf(fun,x0,ntheta,seminfcon)
x = fseminf(fun,x0,ntheta,seminfcon,A,b)
x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq)
x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub)
x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub,options)
x = fseminf(problem)
[x,fval] = fseminf(...)
[x,fval,exitflag] = fseminf(...)
[x,fval,exitflag,output] = fseminf(...)
[x,fval,exitflag,output,lambdas] = fseminf(...)

```

在 `fseminf` 函数调用中，`ntheta` 为 $K_i(w, x)$ 约束条件的个数，`seminfcon` 函数用来定义 $K_i(w, x)$ 和非线性约束条件，返回非线性不等式和等式约束以及 K_i 的大小。`fseminf` 函数退出标志 `exitflag` 取值及相应的含义描述见表 10-11。在使用 `fseminf` 函数求解“半无限”多元函数优化问题中，

需要定义目标函数 fun、 $K_i(w, x)$ 和非线性约束函数 seminfcon。

表 10-11 fminimax 函数 exitflag 标志及含义

exitflag 标志	含 义
1	目标函数收敛于最优解
4	搜索方向幅值小于给定容差或者约束违背小于给定容差 TolCon
5	方向导数幅值小于给定容差或者约束违背小于给定容差 TolCon
0	迭代次数超过最大迭代次数 MaxIter 或者函数计算次数超过给定的 MaxFunEvals
-1	算法因为输出函数终止
-2	无可行解

【例 10-16】求解“半无限”多元约束优化问题。

$$\begin{aligned} & \max 10x_1 + 4.4x_2^2 + 2x_3 \\ & s.t. \begin{cases} 0.5x_3^2 - x_2^2 \geq 3 \\ x_1 + 4x_2 + 5x_3 \leq 32 \\ x_1 + 3x_2 + 2x_3 \leq 29 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases} \end{aligned}$$

其中，半无限约束 $K_i(x, w)$ 为

$$K_1(x, w_1) = \sin(w_1 x_1) \cos(w_1 x_2) = \frac{1}{100} (w_1 - 50)^2 - \sin(w_1 x_3) - x_3 \leq 1 \quad (1 \leq w_1 \leq 100)$$

$$K_2(x, w_2) = \sin(w_2 x_2) \cos(w_2 x_1) = \frac{1}{100} (w_2 - 50)^2 - \sin(w_2 x_3) - x_3 \leq 1 \quad (1 \leq w_2 \leq 100)$$

对于“半无限”多元约束优化问题的求解，首先需要建立目标函数的 M 文件、半无限约束条件及非线性约束的 M 文件如下：

% 目标函数 M 文件

```
function feval=li10_16A(x)
```

```
feval=10*x(1)+4.4*x(2)^2+2*x(3);
```

```
feval=-feval;
```

% 半无限约束条件及非线性约束条件的 M 文件

```
function [c,ceq,K1,K2,s]=li10_16B(x,s)
```

```
if isnan(s(1,1)),
```

```
    s=[0.2 0;0.2 0];
```

```
end
```

```
% 采样的数据点
```

```
w1=1:s(1,1):100;
```

```
w2=1:s(2,1):100;
```

```
% 半无限大约束
```

```
K1=sin(w1*x(1)).*cos(w1*x(2))-1/1000*(w1-50).^2-sin(w1*x(3))-x(3)-1;
```

```
K2=sin(w2*x(2)).*cos(w2*x(1))-1/1000*(w2-50).^2-sin(w2*x(3))-x(3)-1;
```

```
% 非线性约束
```

```
c=3-0.5*x(3)^2-x(2)^2;
```

```
ceq=[];
```

其实现的 MATLAB 程序代码如下：

```
>> clear all;
A=[1 4 5;1 3 2];
b=[32;29];
Aeq=[];
beq=[];
lb=zeros(1,size(A,2));
x0=ones(size(A,2),1);
[x,fval]=fseminf(@li10_16A,x0,2,@li10_16B,A,b,Aeq,beq,lb)
```

运行程序，输出如下：

```
x =
    4.2302
    5.0559
    1.5093
fval =
   -157.7923
```

10.7.2 多目标函数优化

在实际工程应用中，对于一个设计系统的评价和衡量通常不止一个评价指标，如对于电机设计而言，给定一组电机设计参数后，通常要求电机启动转矩大、启动电流小、电机气隙磁场谐波含量低、电机运行效率高等一系列的性能指标。这种含有多个不同的优化目标的问题称为多目标规划问题。在运筹学中，多目标规划问题属于比较复杂的一类优化问题，通常没有固定的求解算法，而且对于求解结果，也不容易评价其优化性能。

多目标规划问题可以描述为

$$\begin{aligned} \min & \{z_1 = f_1(z), z_2 = f_2(z), \dots, z_m = f_m(x)\} \\ \text{s.t.} & \begin{cases} g_i(x) \leq 0 \\ i = 1, 2, \dots, q \end{cases} \end{aligned}$$

定义决策空间 $S = \{x \in R^n \mid g_i(x) \leq 0, i = 1, 2, \dots, q\}$ 、 $Z = \{z \in R^m \mid z_1 = f_1(z), z_2 = f_2(z), \dots, z_m = f_m(x)\}$ 为判据空间，那么 Pareto 最优解集 Ω （非支配解集）定义如下：

$x^0 \in S$ 且 $x^0 \in \Omega \Leftrightarrow$ 不存在其他点 $x \in S$ ，对于 $(1, 2, \dots, m)$ 中的某些 k ， $f_k(x) < f_k(x^0)$ ；对于 $(1, 2, \dots, m)$ 中的某些 l ， $f_l(x) < f_l(x^0)$ 。

而对于无约束的多目标优化问题，Pareto 最优解集 Ω 定义如下：

$z^0 \in \Omega \Leftrightarrow$ 不存在 $z \in Z$ ，对于 $(1, 2, \dots, m)$ 中的某些 k ， $z_k < z_k^0$ ；对于 $(1, 2, \dots, m)$ 中的某些 l ， $z_l < z_l^0$ 。

求解多目标问题，就是要尽可能全面地寻找 Pareto 最优解集。多目标优化问题的处理方法包括以下四种：

(1) 约束法。即确定目标函数的取值范围后，将其转化成约束条件。

(2) 权重法。即将每个目标函数分配一定的权重，进行加权，转化为单目标优化问题。权重法包括固定权重法、适应性权重法和随机权重法。

(3) 目标规划法。通过引入目标函数极值与目标的正偏差和负偏差，将求目标函数的极值问题转化为所有目标函数与对应的目标偏差的最小值问题，进行目标规划求解。

(4) 现代人工智能算法。包括遗传算法、粒子群算法等直接进行多目标优化。

在 MATLAB 优化工具箱中，使用 `fgoalattain` 函数进行多目标优化问题求解，MATLAB 优化工具箱 `fgoalattain` 函数求解多目标优化问题的标准形式为

$$\begin{aligned} \min_{x, \gamma} & F(x) - w\gamma \leq \text{goal} \\ \text{s.t.} & \begin{cases} c(x) \leq 0 \\ c_{\text{eq}} = 0 \\ A \cdot x \leq b \\ A_{\text{eq}} \cdot x = b_{\text{eq}} \\ lb \leq x \leq ub \end{cases} \end{aligned}$$

式中： w 为权重； γ 和 x 同为优化变量；`goal` 为目标函数需要达到的目标向量。多目标规划问题转化为通过优化 γ 和 x ，使目标函数尽可能接近 `goal`。其调用格式如下：

```
x = fgoalattain(fun,x0,goal,weight)
x = fgoalattain(fun,x0,goal,weight,A,b)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon,... options)
x = fgoalattain(problem)[x,fval] = fgoalattain(...)
[x,fval,attainfactor] = fgoalattain(...)
[x,fval,attainfactor,exitflag] = fgoalattain(...)
[x,fval,attainfactor,exitflag,output] = fgoalattain(...)
[x,fval,attainfactor,exitflag,output,lambda] = fgoalattain(...)
```

`fgoalattain` 函数中返回变量 `attainfactor` 表示目标函数与给定目标的偏差程度。如果 `attainfactor` 为正，则表示目标函数值小于给定目标，为负偏差；如果 `attainfactor` 为负，则表示目标函数值大于给定目标，为正偏差。

`fgoalattain` 函数退出标志 `exitflag` 的取值及相应含义描述见表 10-12。

表 10-12 `fgoalattain` 函数 `exitflag` 标志及含义

exitflag 标志	含 义
1	目标函数收敛于最优解
4	搜索方向幅值小于给定容差或者约束违背小于给定容差 TolCon
5	方向导数幅值小于给定容差或者约束违背小于给定容差 TolCon
0	迭代次数超过最大迭代次数 MaxIter 或者函数计算次数超过给定的 MaxFunEvals
-1	算法因为输出函数终止
-2	无可行解

【例 10-17】求解以下的多目标最小化问题。

$$\dot{x} = (A + BKC)x + Bu$$

$$y = Cx$$

使用 MATLAB 优化工具箱 fgoalattain 函数进行多目标求解, 必须给出优化函数的目标向量。首先需要将目标函数转化为标准形式, 得到系数矩阵 A 、 B 、 C 如下:

$$A = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

在这里, 选择优化目标向量为 $(-5, -3, -1)$, 首先建立目标函数的 M 文件:

```
function F = li10_17(K,A,B,C)
```

```
F = sort(eig(A+B*K*C))
```

其实现的 MATLAB 程序代码如下:

```
>> clear all;
A = [-0.5 0 0; 0 -2 10; 0 1 -2];
B = [1 0; -2 2; 0 1];
C = [1 0 0; 0 0 1];
K0 = [-1 -1; -1 -1]; % 初始化控制器矩阵
goal = [-5 -3 -1]; % 目标向量
weight = abs(goal)
lb = -4*ones(size(K0));
ub = 4*ones(size(K0));
options = optimset('Display','iter');
[K,fval,attainfactor] = fgoalattain(@(K)li10_17(K,A,B,C),...
    K0,goal,weight,[],[],[],lb,ub,[],options)
```

运行程序, 其迭代过程如下:

Attainment	Max	Line search	Directional		
Iter F-count	factor	constraint	steplength	derivative	Procedure
0 6		0	1.88521		
1 12	1.031		0.02998	1	0.745
2 18	0.3525		0.06863	1	-0.613
3 24	-0.1706		0.1071	1	-0.223
Hessian modified					
4 30	-0.2236		0.06654	1	-0.234
Hessian modified twice					
5 36	-0.3568		0.007894	1	-0.0812
6 42	-0.3645		0.000145	1	-0.164
Hessian modified					
7 48	-0.3645		0	1	-0.00515
Hessian modified					
8 54	-0.3675		0.0001549	1	-0.00812
Hessian modified twice					
9 60	-0.3889		0.008326	1	-0.0075
Hessian modified					
10 66	-0.3862		0	1	0.00568
11 72	-0.3863	4.193e-013		1	-0.998
Hessian modified twice					

其运行结果如下：

```
K =
-4.0000    -0.2564
-4.0000    -4.0000
fval =
-6.9313
-4.1588
-1.4099
attainfactor =
-0.3863
```

很显然，对于多目标问题的求解，单纯使用 MATLAB 优化工具箱很难取得较好的求解结果，综合使用其他的优化算法和现代人工智能算法，可以使目标优化问题求解变得更加精确。

10.8 动态规划

动态规划（Dynamic Programming）是求解决策过程最优化的有效数学方法，它根据“最优决策的任何截断仍是最优的”这一最优性原理，通过将多阶段决策过程转化为一系列单阶段问题，逐个求解的优化求解方法。自 1957 年 R.E.Bellman 教授出版动态规划的经典专著《Dynamic Programming》以来，动态规划已在经济管理、生产调度、工程技术和最优控制等众多领域得到广泛的应用。逆序算法是动态规划计算的重要算法。MATLAB 中的优化工具箱已被许多设计研究部门和科研工作者使用，成为决策系统的优化计算和设计的有力工具，但该工具箱中尚无动态规划计算的程序文档。

10.8.1 动态规划的概念

1. 基本思想与逆序解法的直观回顾

前面已简单介绍了动态规划，为进一步了解动态规划的基本思想、描述方式和逆序解法，下面来看一个确定网络最短路径问题的例子。

【例 10-18】最短路线问题。

图 10-3 是一个线路网，连线上的数字表示两点之间的距离（或费用）。试寻求一条由 A 到 E 距离最短（或费用最省）的路线。

将该问题划分为 4 个阶段的决策问题，即第 1 阶段为 A 到 $B_j (j=1,2,3)$ ，有 3 种决策方案可供选择；第 2 阶段为从 B_j 到 $C_j (j=1,2,3)$ ，也有 3 种方案可供选择；第 3 阶段阶段为从 C_j 到 $D_j (j=1,2,3)$ ，有 2 种方案可供选择；第 4 阶段为从 D_j 到 E，只有 1 种方案选择。如果用完全枚举法，则可供选择的路线有 $3 \times 3 \times 2 \times 1 = 18$ （条），将其一一比较才可找出最短路线：

$$A \rightarrow B_1 \rightarrow C_2 \rightarrow D_3 \rightarrow E$$

其长度为 12。

显然，这种方法是不经济的，特别是当阶段很多、各阶段可供的选择也很多时，这种解法甚至在计算机上完成也是不现实的。

由于考虑的是从全局上解决求 A 到 E 的最短路问题，而不是就某一阶段解决最短路线，因此可以考虑从最后一阶段开始计算，由后向前逐步推至 A 点。

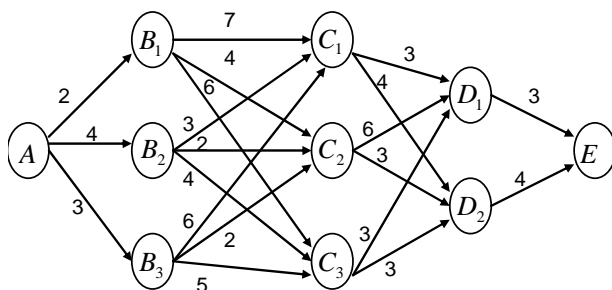


图 10-3 线路图

第4阶段, 由 D_1 到 E 只有一条路线, 其长度 $f_4(D_1)=3$, 同理 $f_4(D_2)=4$ 。

第3阶段, 由 C_j 到 D_i 分别均有2种选择, 即 $f_3(C_1)=\min\{C_1D_1+f_4(D_1)+C_1D_2+f_4(D_2)\}=\min\{C_3D_1+f_4(D_1), C_3D_2+f_4(D_2)\}=\min\{3+3, 3+4\}=6$, 决策点为 D_1 。

$$f_3(C_2)=\min\{C_2D_1+f_4(D_1), C_2D_2+f_4(D_2)\}=\min\{6+3, 3+4\}=7$$

$$f_3(C_3)=\min\{C_3D_1+f_4(D_1), C_3D_2+f_4(D_2)\}=\min\{3+3, 3+4\}=6$$

第2阶段, 由 B_j 到 C_i 分别均有3种选择, 即:

$f_2(B_1)=\min\{B_1C_1+f_3(C_1), B_1C_2+f_3(C_2), B_1C_3+f_3(C_3)\}=\min\{7+6, 4+7, 6+6\}=11$, 决策点为 C_2 。

$f_2(B_2)=\min\{B_2C_1+f_3(C_1), B_2C_2+f_3(C_2), B_2C_3+f_3(C_3)\}=\min\{3+6, 2+7, 4+6\}=9$, 决策点为 C_1 或 C_2 。

$f_2(B_3)=\min\{B_3C_1+f_3(C_1), B_3C_2+f_3(C_2), B_3C_3+f_3(C_3)\}=\min\{6+6, 2+7, 5+6\}=9$, 决策点为 C_2 。

第1阶段, 由 A 到 B , 有3种选择, 即:

$f_1(A)=\min\{AB_1+f_2(B_1), AB_2+f_2(B_2), AB_3+f_2(B_3)\}=\min\{2+11, 4+9, 3+9\}=12$, 决策点为 B_3 。

$f_1(A)=12$ 说明从 A 到 E 的最短距离为 12, 最短路线的确定可按计算顺序反推而得。即

$$A \rightarrow B_1 \rightarrow C_2 \rightarrow D_3 \rightarrow E$$

从例 10-18 的求解过程可以得到以下启示:

(1) 对一个问题是否用上述方法求解, 其关键在于能否将问题转化为相互联系的决策过程相同的多个阶段决策问题。多阶段决策问题是: 把一个问题看做是一个前后关联具有链状结构的多阶段过程, 也称为序贯决策过程, 如图 10-4 所示。

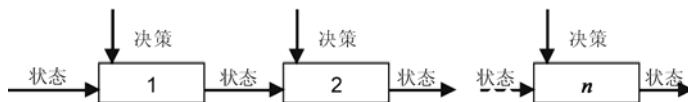


图 10-4 序贯决策过程

(2) 在处理各阶段决策的选取上, 不仅只依赖于当前面临的状态, 而且还要注意对以后的发展。即是从全局考虑解决局部(阶段)的问题。

(3) 阶段选取的决策, 一般与“时序”有关, 决策依赖于当前的状态, 又随即引起状态的转移, 整个决策序列就是在变化的状态中产生出来, 故有“动态”含义。因此, 把这种方法称为动态规划方法。

(4) 决策过程是与阶段发展过程逆向而行。

2. 动态规划的基本概念及其数学描述

1) 阶段

整个问题的解决可分为若干个相互联系的阶段依次进行。通常按时间或空间划分阶段，描述阶段的变量称为阶段变量，记为 k 。

2) 状态

状态表示每个阶段开始所处的自然状况或客观条件，它描述了研究问题过程的状况。各阶段的状态通常用状态变量描述。常用 x_k 表示第 k 阶段的状态变量。 n 个阶段的决策过程有 $n+1$ 个状态。用动态规划方法解决多阶段决策问题时，要求整个过程具有无后效性，即如果某阶段的状态给定，则此阶段以后过程的发展不受以前状态的影响，未来状态只依赖于当前状态。

3) 决策

某一阶段的状态确定后，可以作出各种选择，从而演变到下一阶段某一状态，这种选择手段称为决策。描述决策的变量称为决策变量。决策变量限制的取值范围称为允许决策集合。用 $u_k(x_k)$ 表示第 k 阶段处于状态 x_k 时的决策变量，它是 x_k 的函数，用 $D_k(x_k)$ 表示 x_k 的允许决策的集合。

4) 策略

一个由每个阶段的决策按顺序排列组成的集合称为策略，用 P 表示，即 $p(x_1) = \{u_1(x_1), u_2(x_2), \dots, u_n(x_n)\}$ 。由第 k 阶段的状态 x_k 开始到终止状态的后部子过程的策略记为 $p_k(x_k)$ ，即 $p_k(x_k) = \{u_k(x_k), u_{k+1}(x_{k+1}), \dots, u_n(x_n)\}$ 。在实际问题中，可供选择的策略有一定范围，此范围称为允许策略集合。允许策略集合中达到最优效果的策略称为最优策略。

5) 状态转移方程

如果第 k 个阶段状态变量为 x_k ，作出的决策为 u_k ，那么第 $k+1$ 阶段的状态变量 x_{k+1} 也被完全确定。用状态转移方程表示这种演变规律，写做

$$x_{k+1} = T_k(x_k, u_k)$$

6) 指标函数和最优值函数

指标函数是系统执行某一策略所产生结果的数量表示，是用来衡量策略优劣的数量指标，它定义在全过程和所有后部子过程上，分别用 G 和 G_k 表示。即

$$G(u_1, u_2, \dots, u_n, x_1, x_2, \dots, x_{n+1}) \text{ 和 } G_k(u_k, u_{k+1}, \dots, u_n, x_k, x_{k+1}, \dots, x_{n+1})$$

7) 最优策略和最优轨线

使指标函数 G_k 达到最优值的策略是从阶段 k 开始的后部子过程的最优策略，记为 $p_k^* = \{u_k^*, \dots, u_{k+1}^*, u_n^*\}$ ， p^* 是全过程的最优策略，简称为最优策略。从初始状态 $x_1 (= u_1^*)$ 出发，过程按照 p^* 和状态转移方程演变所经历的状态序列 $x_1^*, \dots, x_2^*, x_{n+1}^*$ 称为最优轨线。

10.8.2 逆序算法及 MATLAB 的实现

1. 逆序算法的基本方程

由例 10-18 的求解过程可以看出，下面的方程在动态规划逆序求解中起着本质的作用：

$$\begin{cases} f_k(x_k) = \min\{G_k(v_k(x_k, u_k(x_k))), f_{k+1}(x_{k+1})\} | u_k \in D_k(x_k) \\ f_{n+1}(x_{n+1}) = 0, x_{k+1} = T_k(x_k, u_k), k = n, n-1, \dots, 1 \end{cases}$$

称此为动态规划逆序求解的基本方程。注意，在例 10-18 中，有

$$G_k(v_k(x_k, u_k), f_{k+1}(x_{k+1})) = v_k(x_k, u_k) + f_{k+1}(x_{k+1})$$

可以把建立动态规划模型归纳成以下几个步骤:

- (1) 将问题恰当地划分为若干个阶段。
- (2) 正确选择状态变量, 使它既能描述过程的演变, 又满足无后效性。
- (3) 规定决策变量, 确定每个阶段的允许决策集合。
- (4) 写出状态转移方程。
- (5) 确定各阶段各种决策的阶段指标, 列出计算各阶段最优后部策略指标的基本方程。

2. 动态规划逆序算法的 MATLAB 程序

```
function [p_o,fval]=dynprog(x,dec,sub,tran,obj)
% 自由始端和终端的动态规划,求指标函数最小值的逆序(或后向)算法递归计算程序
% x 是状态变量,一列代表一个阶段状态;M 函数 dec(k,x)由阶段 k 的状态变量 x 求出相应
% 的允许决策变量;M 函数 sub(k,x,u)是阶段指标函数,M 函数 tran(k,x,u)是状态转移函数,
% 其中 x 是阶段 k 的状态变量,u 是相应的决策变量;M 函数 obj(v,f)是第 k 阶段直至最后阶段的
% 指标函数,当 obj(v,f)=v+f 时,obj(v,f)可以省略
% 输出 p_o 由 4 列构成,p_o=[序列号;最优策略组;最优轨线组;指标函数值组];
% fval 是一个列向量,各元素分别表示 p_o 各最优策略组对应始端状态 x 的最优函数值
k=length(x(1,:));
x_isnan=~isnan(x);
t_vub=inf;
t_vubm=inf*ones(size(x));
f_o=nan*ones(size(x));
d_o=f_o;
% 计算终端(最后一阶段)相关值
tmp1=find(x_isnan(:,k));
tmp2=length(tmp1);
for i=1:tmp2
    u=feval(dec,k,x(i,k));
    tmp3=length(u);
    for j=1:tmp3
        tmp=feval(sub,k,x(tmp1(i),k),u(j));
        if tmp<=t_vub,
            f_o(i,k)=tmp;
            d_o(i,k)=u(j);
            t_vub=tmp;
        end
    end
end
end
for ii=k-1:-1:1
    tmp10=find(x_isnan(:,ii));
    tmp20=length(tmp10);
    for i=1:tmp20
        u=feval(dec,ii,x(i,ii));
        tmp30=length(u);
        for j=1:tmp30
            tmp00=feval(sub,ii,x(tmp10(i),ii),u(j));
            tmp40=feval(tran,ii,x(tmp10(i),ii),u(j));
```

```

        tmp50=x(:,ii+1)-tmp40;
        tmp60=find(tmp50==0);
        if ~isempty(tmp60),
            if nargin<5,
                tmp00=tmp00+f_o(tmp60(1),ii+1);
            else
                tmp00=feval(obj,tmp00,f_o(tmp60(1),ii+1));
            end
            if tmp00<=t_vubm(i,ii)
                f_o(i,ii)=tmp00;
                d_o(i,ii)=u(j);
                t_vubm(i,ii)=tmp00;
            end
        end
    end
end
end
fval=f_o(tmp1,1);
fval=fval(find(~isnan(fval)),1);
% 记录最优决策,最优轨线和相应指标函数值
p_o=[];tmpx=[];
tmpd=[];tmpf=[];
tmp0=find(x_isnan(:,1));
tmp01=length(tmp0);
for i=1:tmp01
    tmpd(i)=d_o(tmp0(i),1);
    tmpx(i)=x(tmp0(i),1);
    tmpf(i)=feval(sub,1,tmpx(i),tmpd(i));
    p_o(k*(i-1)+1,[1,2,3,4])=[1,tmpx(i),tmpd(i),tmpf(i)];
    for ii=2:k
        tmpx(ii)=feval(tran,ii-1,tmpx(i),tmpd(i));
        tmp1=x(:,ii)-tmpx(ii);
        tmp2=find(tmp1==0);
        if ~isempty(tmp2)
            tmpd(ii)=d_o(tmp2(1),ii);
        end
        tmpf(ii)=feval(sub,ii,tmpx(ii),tmpd(ii));
        p_o(k*(i-1)+ii,[1,2,3,4])=[ii,tmpx(ii),tmpd(ii),tmpf(ii)];
    end
end
end

```

3. Dijkstra 算法的 MATLAB 程序

function [S,D]=minRoute(i,m,W,opt)

% 图与网络论中求最短路径的 Dijkstra 算法 M 函数格式[S,D]=minRoute(i,m,W,opt)

% i 为最短路径的起始点, m 为图顶点数, W 为图的带权邻接矩阵, 不构成边的两顶点之间的
% 权重用 inf 表示。S 的每一列从上到下记录了从始点到终点的最短路径所经顶点的序号。opt=0(默
% 认值)时, S 按终点序号从小到大显示结果; opt=1 时, S 按最短路径从小到大显示结果。D 是
% 一行向量, 对应记录了 S 各列所示路径的大小

```

if nargin<4
    opt=0;
end
dd=[];tt=[];
ss=[];ss(1,1)=i;
V=1:m;V(i)=[];
dd=[0;i];
kk=2;
[mdd,ndd]=size(dd);
while ~isempty(V)
    [tmpd,j]=min(W(i,V));
    tmpj=V(j);
    for k=2:ndd
        [tmp1,jj]=min(dd(1,k)+W(dd(2,k),V));
        tmp2=V(jj);
        tt(k-1,:)= [tmp1,tmp2,jj];
    end
    tmp=[tmpd,tmpj,j;tt];
    [tmp3,tmp4]=min(tmp(:,1));
    if tmp3==tmpd
        ss(1:2,kk)= [i,tmp(tmp4,2)];
    else
        tmp5=find(ss(:,tmp4)~=0);
        tmp6=length(tmp5);
        if dd(2,tmp4)==ss(tmp6,tmp4)
            ss(1:tmp6+1,kk)= [ss(tmp5,tmp4);tmp(tmp4,2)];
        else
            ss(1:3,kk)= [i;dd(2,tmp4);tmp(tmp4,2)];
        end
    end
    end
    dd=[dd,[tmp3;tmp(tmp4,2)]];
    V(tmp(tmp4,3))=[];
    [mdd,ndd]=size(dd);
    kk=kk+1;
end
if opt==1
    [tmp,t]=sort(dd(2,:));
    S=ss(:,t);
    D=dd(1,t);
else
    S=ss;
    D=dd(1,:);
end
end

```

10.8.3 动态规划的应用

1. 最短路径问题

【例 10-19】求图 10-5 中，分别自点 v_1 和 v_3 到其他各点的最短有向路径。

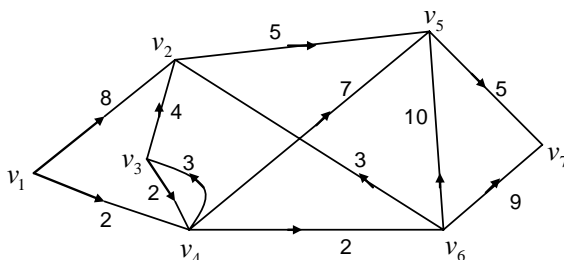


图 10-5 有向图

其实现的 MATLAB 程序代码如下：

```
>> clear all;
w=inf*ones(7);
w(1,[2,4])=[8,2];
w(2,5)=5;
w(3,[2,4])=[4,2];
w(4,[3,5,6])=[3,7,2];
w(5,7)=5;
w(6,[2,5,7])=[7,10,9];
[s1,d1]=minroute(1,7,w,1)
[s3,d3]=minroute(3,7,w,1)
```

运行程序输出如下：

```
s1 = %输出，每列表示最短路径的顶点序号
    1    1    1    1    1    1    1
    0    2    4    4    4    4    4
    0    0    3    0    5    6    6
    0    0    0    0    0    0    7
d1 = %最短路径的权值
    0    8    5    2    9    4    13
s3 = %输出，每列表示最短路径的顶点序号
    3    3    3    3    3    3    3
    1    2    0    4    4    4    4
    0    0    0    0    5    6    6
    0    0    0    0    0    0    7
d3 = %最短路径的权值
    Inf    4    0    2    9    4    13
```

2. 生产计划问题

【例 10-20】某工厂与用户订立合同，在四个月内出售一定数量的某种产品，产量限制为 10 的倍数，工厂每月最多生产 100 件，产品可以存储，存储费用为每台 200 元，每个月的需求量及每件产品的生产成本见表 10-13。

表 10-13 每个月的需求量及每件产品的生产成本

月 份	每件产品生产成本/百元	需要量/件
1	70	60
2	72	70
3	80	120
4	76	60

现在分别在如下两种情况下确定每月的生产量,要求既能满足每月的合同需求量,又使生产成本和存储费用最低。

(1) 1月初没有存货可用。

(2) 1月初有 20 件存货可用。

静态地看,这是一个整数规划问题,但可以把这个问题的解决动态地视为各月(阶段)先后作出决策(这里指生产量)的过程——多阶段的决策过程,而在每个月作决策时,不能仅考虑本月的费用(阶段指标),因为本月的决策会对以后各月的决策产生影响,因此应考虑从本月直到第4月末的总费用(总指标),而每月的决策依赖于各月初仓库中的存货量(始端),而和以前各月如何造成该存货量的情况无关(无后效性)。

这是一个4阶段动态规划问题。如果用逆序法解题,第1阶段是1月份,……,第4阶段是4月份。

阶段指标的函数为

$$v_k(x_k, u_k) = c_k u_k + 2x_k$$

状态转移方程为

$$x_{k+1} = x_k + u_k - q_k$$

基本方程为

$$\begin{cases} f_4(x_4, u_4) = u_4(x_4, u_4) \\ f_k(x_k, u_k) = \min\{u_k(x_k, u_k) + f_{k+1}(x_{k+1}) | u_k \in D_k(x_k)\}, k = 3, 2, 1 \end{cases}$$

式中: x_k 为第 k 阶段开始的产品存储数(状态变量); u_k 为第 k 阶段的产量(决策变量); c_k 为第 k 阶段每件产品的生产成本; q_k 为第 k 阶段的需求量。

对于本例的问题(1),1月初没有存货可用,可首先分析出各月的最大存储量和产量。如对4月初,前3个月的最大产量为300件(每月最大产量为100件),实际需求量为 $60+70+120=250$ 件,所以4月初的最大存储量为 $300-250=50$ 件。因产量限制为10的倍数,4月初的存储量只可能是0、10件、20件、30件、40件、50件这六种情况。而4月份的实际需要量为60件,因此第4阶段的产量(决策)相应为60件、50件、40件、30件、20件、10件,进而计算 $f_4(x_4, u_4)$ 。再分析3月初情况,等等。如此可仿照例10-18逐段手工计算,求出最优决策。

下面将调用自定义编写的 `dynprog.m` 进行计算。由于计算机的优势,这里将就1月初存货分别为0、10件、20件、30件、40件、50件、60件的所有可能情况进行计算。把此问题作为自由始端动态规划考虑,此时各阶段的最大存货出现在 $x_1=60$ 时,经分析可知 $0 \leq x_2 \leq 100$, $0 \leq x_3 \leq 130$, $0 \leq x_4 \leq 60$,考虑到产量是10的倍数,根据上面所述的阶段指标函数、状态转移方程和基本方程,写出下面的3个M函数以备计算时调用。

% 自定义编写的 `li10_20A.m` 文件

```
function u=li10_20A(k,x)
```

% 在阶段 k 由状态变量 x 的值求出其相应的决策变量所有的取值

```
c=[70,72,80,76];
```

```
q=10*[6,7,12,6];
```

```
if q(k)-x<0,
```

```
    u=0:100;      % 决策变量不能取为负值
```

```
else
```

```
    u=q(k)-x:100;    % 产量满足需求且不超过 100
```

```
end
```

```
u=u(k);
```

% 自定义编写的 li10_20B.m 文件

```
function v=li10_20B(k,x,u)
```

% 阶段 k 的指标函数

```
c=[70,72,80,76];
```

```
v=c(k)*u+2*x;
```

% 自定义编写的 li10_20C.m 文件

```
function y=li10_20C(k,x,u)
```

% 状态转移方程

```
q=10*[6,7,12,6];
```

```
y=x+u-q(k);
```

其实现的 MATLAB 程序代码如下：

```
>> clear all;
```

```
x=nan*ones(14,4);    %x 是 10 的倍数,最大范围为  $0 \leq x \leq 130$ 
```

```
x(1:7,1)=10*(0:6)';    %按月定义 x 的可能取值
```

```
x(1:11,2)=10*(0:10)';
```

```
x(1:12,3)=10*(2:13)';
```

```
x(1:7,4)=10*(0:6)';
```

```
[p,f]=dynprog(x,@li10_20A,@li10_20B,@li10_20C)
```

运行程序，输出如下：

```
p =
```

1	0	100	7000
2	40	100	7280
3	70	50	4140
4	0	60	4560
1	10	100	7020
2	50	100	7300
3	80	40	3360
4	0	60	4560
1	20	100	7040
2	60	100	7320
3	90	30	2580

4	0	60	4560
1	30	100	7060
2	70	100	7340
3	100	20	1800
4	0	60	4560
1	40	100	7080
2	80	100	7360
3	110	10	1020
4	0	60	4560
1	50	100	7100
2	90	100	7380
3	120	0	240
4	0	60	4560
1	60	100	7120
2	100	100	7400
3	130	0	260
4	10	50	3820

f =

22980
22240
21500
20760
20020
19280
18600

由 p 的第 1 行和第 3 行可以看出 1 月初无存货和有存货 20 件的最优决策, 现将其列成表 10-14 所示。

表 10-14 分析结果

1 月初的 存货量/件	阶段序号 (月份)	最优轨线 (储存量)	最优决策 (产量)/件	阶段指标函数 数值/百元
0	1	0	100	7000
	2	40	100	7280
	3	70	50	4140
	4	0	60	4560
20	1	20	100	7040
	2	60	100	7320
	3	90	30	2580
	4	0	60	4560

由 f 的第 1、3 行可以看出对应的 4 个月的总成本分别为 22980 元和 21500 元。

3. 资源分配问题

【例 10-21】某机构根据国家计划的安排, 拟将某种高效率的设置 5 台分配给所属的甲、乙、丙 3 个工厂, 各工厂若获得这种设备之后, 可以为国家提供的盈利见表 10-15。问这 5 台设备

如何分配才能使国家得到的盈利最大？

表 10-15 3 个工厂可以为国家提供的盈利表

设备数	工厂		
	甲	乙	丙
0	0	0	0
1	3	5	4
2	7	10	6
3	9	11	11
4	12	11	12
5	13	11	13

将问题按工厂分为 3 个阶段，甲、乙和丙 3 个工厂分别编号为 1、2 和 3。

设状态变量 x_k 表示分配给第 k 个工厂至第 n 个工厂的设备台数。决策变量 u_k 表示分配给第 k 个工厂的设备台数。则状态转移方程 $x_{k+1} = x_k - u_k$ ， x_{k+1} 为分配给第 $k+1$ 个工厂至第 n 个工厂的设备台数。

设阶段指标函数 $v_k(x_k, u_k)$ 表示 u_k 台设备分配到第 k 个工厂所获得的盈利值。 $f_k(x_k)$ 表示 x_k 台设备分配给第 k 个工厂至第 n 个工厂所获得的最大盈利值。

则基本方程为

$$\begin{cases} f_k(x_k) = \max\{v_k(u_k) + f_{k+1}(x_{k+1}) | u_k\}, k = 2, 1 \\ f_3(x_3) = v_3(u_3) \end{cases}$$

利用计算机计算的优势，可根据表 10-15 将本问题视为自由始端，即初始状态 $x=0, 1, 2, 3, 4, 5$ 的动态规划问题求解。

自定义编写 3 个 M 函数，以借计算时调用。

% 自定义编写的 li10_21A.m 文件

```
function u=li10_21A(k,x)
```

```
function u=li10_21A(k,x)
```

% 在阶段由状态变量 x 的值求出其相应的决策变量所有的取值

if $k==3$, % 第三阶段丙工厂得到所有剩余设备

```
u=x;
```

else

```
u=0:x;
```

```
end
```

% 自定义编写的 li10_21B.m 文件

```
function v=li10_21B(k,x,u)
```

% 阶段 k 的指标函数

```
w=[0 0 0;3 4 5;7 10 6;9 11 11;12 11 12;13 11 12];
```

$w=-w$; % 标准化,即将基本方程中 \max 转化为 \min

```
v=([0 1 2 3 4 5]==u)*w(:,k); %对此题可定义 v=w(u+1,k);
```

% 自定义编写的 li10_21C.m 文件

```
function y=li10_21C(k,x,u)
```

```
y=x-u; %状态转移方程
```

其实现的 MATLAB 程序代码如下:

```
>> clear all;
x=[0;1;2;3;4;5];
x=[x,x,x];
[p,f]=dynprog(x,@li10_21A,@li10_21B,@li10_21C)
```

运行程序, 输出如下:

```
p =
     1     0     0     0
     2     0     0     0
     3     0     0     0
     1     1     0     0
     2     1     0     0
     3     1     1    -5
     1     2     0     0
     2     2     2   -10
     3     0     0     0
     1     3     0     0
     2     3     2   -10
     3     1     1    -5
     1     4     1    -3
     2     3     2   -10
     3     1     1    -5
     1     5     2    -7
     2     3     2   -10
     3     1     1    -5

f =
     0
    -5
   -10
   -15
   -18
   -22
```

由 p 和 f 可见, 有 5 台设备时可分配给甲、乙和丙工厂分别各 2 台、2 台和 1 台设备, 获最大盈利 21; 有 4 台设备时可分配给甲、乙和丙分别为 2 台、2 台和 0 台设备, 获最大盈利 17; 如此也可知有 3 台、2 台、1 台、0 台设备时的最优分配方案。

4. 任务均衡问题

【例 10-22】现有 4 种不同的车床 1, 2, 3, 4, 同时加工 500 件相同的零件。各车床加工一个零件的时间分别为 0.5h、0.1h、0.2h 和 0.05h, 问如何给 4 个车床分配加工零件数目, 使完工时间最短?

将问题按车床编号分为 4 个阶段。设状态变量 x_k 表示分配给第 k 号车床至第 4 号车床的零件数，决策变量 u_k 表示分配给第 k 号车床的零件数， $u_k = 0, 1, \dots, x_k$ ，则状态转移方程 $x_{k+1} = x_k - u_k$ ，阶段指标函数 $v_k(u_k)$ 表示 u_k 个零件分配到第 k 号车床加工所需的时间， $v_k(u_k) = u_k t_k$ ， t_k 是 k 号车床的加工时间。 $f_k(x_k)$ 表示 x_k 个零件分配给第 k 至第 4 号车床加工所需的最短时间。则基本方程为

$$\begin{cases} f_k(x_k) = \min\{G * v_k(u_k) + f_{k+1}(x_{k+1}) | u_k\}, k = 4, 3, 2, 1 \\ f_5(x_5) = 0, G(a, b) = \max(a, b) \quad (\text{用最长的车床所需时间为总加工时间}) \end{cases}$$

将本问题视为初始状态 $x=500$ 的动态规划问题求解。

根据上面所述的阶段指标函数、状态转移方程和基本方程，自定义编写 4 个 M 函数以备计算时调用。

% 自定义编写 li10_22A 函数

```
function u=li10_22A(k,x)
```

```
u=0:x;
```

% 自定义编写 li10_22B 函数

```
function v=li10_22B(k,x,u)
```

```
t=[0.5;0.1;0.2;0.05];
```

```
if k==5
```

```
    v=0;
```

```
else
```

```
    v=u*t(k);
```

```
end
```

% 自定义编写 li10_22C 函数

```
function y=li10_22C(k,x,u)
```

```
y=x-u;
```

% 自定义编写 li10_22D 函数

```
function y=li10_22D(v,f)
```

```
y=max(v,f);
```

其实现的 MATLAB 程序代码如下：

```
>> clear all;
n=500;
x1=[n;nan*ones(n,1)];
x2=0:n;
x3=[0;nan*ones(n,1)];
x=[x1,x2',x2',x3];
[p,f]=dynprog(x,@li10_22A,@li10_22B,@li10_22C,@li10_22D)
```

运行程序，输出如下：

p =

1.0000	500.0000	27.0000	13.5000
2.0000	473.0000	135.0000	13.5000
3.0000	338.0000	67.0000	13.4000
4.0000	271.0000	271.0000	13.5500
5.0000	0	0	0

f =

13.5500

参 考 文 献

- [1] 许波, 刘征. MATLAB 工程数学应用. 北京: 清华大学出版社, 2000.
- [2] 贺子兴, 等. 概率论与数理统计. 北京: 科学出版社, 2000.
- [3] 刘卫国. 科学计算与 MATLAB 语言. 北京: 中国铁道出版社, 2000.
- [4] 陆同兴. 非线性物理概论. 合肥: 中国科学技术大学出版社, 2002.
- [5] 张智星. MATLAB 程序设计与应用. 北京: 清华大学出版社, 2002.
- [6] 王松桂, 等. 概率论与数理统计. 北京: 科学出版社, 2002.
- [7] 魏巍. MATLAB 应用数学工具箱技术手册. 北京: 国防工业出版社, 2002.
- [8] 赵选民, 等. 数学统计. 北京: 科学出版社, 2002.
- [9] 于义良, 张银生. 实用概率统计. 北京: 中国人民大学出版社, 2002.
- [10] 梅常林, 周家良. 实用统计方法. 北京: 科学出版社, 2002.
- [11] 何强, 何英. MATLAB 扩展编程. 北京: 清华大学出版社, 2002.
- [12] 奥特 R L, 朗格内克 M. 统计学方法与数据分析引论. 北京: 科学出版社, 2003.
- [13] 丁杰, 高文杰. 概率统计. 天津: 天津大学出版社, 2004.
- [14] 汤大林, 张玉环, 陈淑敏, 等. 概率论与数理统计. 天津: 天津大学出版社, 2004.
- [15] 薛定宇, 陈阳泉. 高等应用数学问题的 MATLAB 求解. 北京: 清华大学出版社, 2004.
- [16] 苏金明, 王永利. MATLAB 7.0 实用指南: 上册. 北京: 电子工业出版社, 2004.
- [17] 肖伟, 刘忠, 等. MATLAB 程序设计与应用. 北京: 清华大学出版社, 2005.
- [18] 杨振海, 张忠占. 应用数理统计. 北京: 北京工业大学出版社, 2005.
- [19] 高惠旋. 应用多元统计分析. 北京: 北京大学出版社, 2005.
- [20] 王岩, 隋思涟, 王爱青. 数理统计与 MATLAB 工程数据分析. 北京: 清华大学出版社, 2006.
- [21] 李勇, 张淑敏. 统计学导论. 北京: 人民邮电出版社, 2006.
- [22] 庄楚强, 何春雄. 应用数理统计基础. 广州: 华南理工大学出版社, 2006.
- [23] 刘金兰. 管理统计学. 天津: 天津大学出版社, 2007.
- [24] 许国根, 许萍萍. 化学化工中的数学方法与 MATLAB 实现. 北京: 化学工业出版社, 2007.
- [25] 董振海. 精通 MATLAB 7 编程与数据库应用. 北京: 电子工业出版社, 2007.
- [26] 龚纯, 王正林. MATLAB 语言常用算法程序集. 北京: 电子工业出版社, 2008.
- [27] 包研科, 李娜. 数理统计与 MATLAB 数据处理. 沈阳: 东北大学出版社, 2008.
- [28] 许国根, 许萍萍. 化学化工中的数学方法及 MATLAB 实现. 北京: 化学工业出版社, 2008.
- [29] 龙脉工作室, 刘会灯, 朱飞. MATLAB 编程基础与典型应用. 北京: 人民邮电出版社, 2008.
- [30] 庞中华, 崔红. 系统辨识与自适应控制 MATLAB 仿真. 北京: 北京航空航天大学出版社, 2009.
- [31] 刘正君. MATLAB 科学计算与可视化仿真宝典. 北京: 电子工业出版社, 2009.
- [32] 隋思涟, 王岩. MATLAB 语言与工程数据分析. 北京: 清华大学出版社, 2009.
- [33] 李明雨, 李勇, 蒋宝锋. MATLAB R2008 数学和控制实例教程. 北京: 化学工业出版社, 2009.

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为，歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396；(010) 88258888

传 真：(010) 88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036